THEORY OF COMPUTATION

As per new CBCS Syllabus for VI Sem BCA of Bangalore University

Sarakutty T K
T Kohila Kangalakshmi

SKYWARD PUBLISHERS
A professional team

6010 8637

Theory of Computation

(As per New CBCS Syllabus, for VI Semester BCA, Bangalore University)

By

Sara Kutty.T.K
Assistant Professor
Dayananda Sagar College
Bangalore

&

T. KohilaKanagalakshmi
Assistant Professor
Dayananda Sagar College
Bangalore

SKYWARD PUBLISHERS
A professional team

Skyward Publishers
#157, 7th Cross, 3rd Main Road, Chamarajpet
Bangalore-18. Phone : 080-64553333, 9611185999
E-mail: skyward.publishers@gmail.com
www.skywardpublishers.com

PREFACE

This book is organized in such a way that it covers the entire Theory of Computation Syllabus prescribed by Bangalore University for 6th semester BCA students. The book is primarily designed to provide an introduction to some of the fundamental concepts of automata theory for the under graduate and the post graduate students in the field of computer science. It enables the student to gain knowledge of automata theory with ease and hence there is no need to have any mathematical prerequisites to understand the concepts elucidated in the book. The theory part in each chapter is explained in a lucid and unambiguous manner. A number of examples along with worked out solutions illustrate the theory in each chapter.

Our sincere thanks to management, principal and faculties of Dayananda Sagar College for their encouragement and support.

Our special thanks to Skyward Publishers team for inviting us to prepare and publish this book.

We authors would like to express our gratitude to our family and friends. It is their wishes which makes us more determined. We would also like to thank the almighty for guiding us in all our efforts.

Constructive suggestions from teachers and students to improve the book is always welcome to skyward.publishers@gmail.com.

-Authors

SYLLABUS

1. Introduction to Finite Automata (12 Hrs)

Introduction to Finite Automata: The central concepts of Automata theory; Deterministic finite automata; Nondeterministic finite automata. An application of finite automata, Finite automata with Epsilon transitions.

2. Regular Expression (12 Hrs)

Regular Expressions: Finite Automata and Regular Expressions Applications of Regular Expressions. Regular languages; Proving languages not to be regular languages; Closure properties of regular languages; Decision properties of regular languages; Equivalence and minimization of automata.

3. Context Free Grammars (12 Hrs)

Context-free grammars: Parse trees; Applications; Ambiguity in grammars and Languages. Definition of the Pushdown automata; the languages of a PDA; Equivalence of PDA's and CFG's.

4. Normal Forms for CFG (12 Hrs)

Deterministic Pushdown Automata:Normal forms for CFGs; The pumping lemma for CFGs; Closure properties of CFLs. Problems that Computers cannot solve.

5. Turing Machine (12 Hrs)

The Turing machine:Programming techniques for Turing Machines. Undecidability, A Language that is not recursively enumerable; An Undecidable problem that is RE; Post's Correspondence problem.

CONTENTS

UNIT 1

INTRODUCTION TO FINITE AUTOMATA

CHAPTER OUTLINE

1.0 Introduction

In theoretical computer science the theory of computation is the branch that deals with whether and how efficiently problems can be solved on a model of computation, using an algorithm. In order to perform a rigorous study of computation, computer scientists work with a mathematical abstraction of computers called a model of computation like Finite Automata, Turing machine, etc., Automata theory is the study of abstract machines (i.e., abstract mathematical machines) and the computational problems that can be solved using these machines. These abstract machines are called automata.

1.1 Set

Set is a well defined collection of objects. Individual objects in the set are called members/ elements of the set.

We use the capital letters $A, B, C, \ldots.$ for denoting the sets. The small letters $a, b, c, \ldots.$ are used to denote the elements of any set. When a is an element of the set A, we write $a \in A$. When a is not an element of A. We write $a \notin A$.

Example :

Set of even numbers greater than 0 and less than 10.

$$A = \{2, 4, 6, 8\}$$

Note All the elements of the set should be enclosed between '{' and '}' and separated by commas. The elements of the set can be in any order.

Example:

Set of vowels $A = \{a, e, i, o, u\}$

Subset: Let A and B be two sets, if every element of set A is present in set B then A is called the subset of B, and is denoted by $A \subseteq B$.

Example :

$$A = \{a, b, c\}$$
$$B = \{a, b, c, d, e\} \text{ then } A \subseteq B.$$

Equal Set: Let A and B be two sets. They are called equal sets if both the sets have same members.

Example :

$$A = \{a, b, c\}$$
$$B = \{a, b, c\} \text{ then } A = B$$

Note If $A = B$ then $A \subseteq B$ and $B \subseteq A$

Empty Set or Null Set ($\varnothing$): The set having no elements in it is called empty set. It is denoted by $\emptyset$ or { }

Power Set: Let A be the set. The set of all subsets of set A is called Power set of A and is denoted by 2^A.

Example :

$A = \{a, b, c\}$

Power set, $2^A = \{ \emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\} \{b, c\}, \{a, b, c\} \}$

Operations on Set

(i) **Union ($\cup$):** The Union of two sets A and B is given by $A \cup B = \{ x \mid x \in A \text{ or } x \in B\}$ (i.e.,) combination of both the sets.

Example :

Let

$$\begin{aligned} A &= \{a, b, c, d\} \\ B &= \{c, d, e\} \\ A \cup B &= \{a, b, c, d, e\} \end{aligned}$$

(ii) **Intersection ($\cap$):** The Intersection of two sets A and B is given by

$A \cap B = \{ x \mid x \in A \text{ and } x \in B \}$ (i.e.,) collection of common elements in both the sets A and B.

Example :

Let

$$\begin{aligned} A &= \{a, b, c, d\} \\ B &= \{c, d, e\} \\ A \cap B &= \{c, d\} \end{aligned}$$

Difference

The difference of two sets A and B is given by $A - B = \{x \mid x \in A \text{ or } x \notin B\}$

Example :

Let

$$\begin{aligned} A &= \{a, b, c, d\} \\ B &= \{c, d, e\} \\ A - B &= \{a, b\} \\ B - A &= \{e\} \end{aligned}$$

1.2 Terminologies used in Finite Automata

The word "automata" is used for recognizing. A machine understands a set of words, that is, it accepts a set of words and rejects some others. The study of such machines is known as automata theory.

Symbol: Symbol is an abstract entity.

Example :

Letters : *a, b, c,* ___

Digits : 0, 1, 2, ____

Alphabet (Σ): An alphabet or character set is a finite non-empty set of symbols.

Example :

Binary alphabets, $\Sigma = \{0, 1\}$

English alphabets, $\Sigma = \{a, b, c, \_\_\_\_\_ z\}$

String: A String or word is a finite sequence of symbols from the alphabet, Σ.

Example :

Let $\Sigma = \{0, 1\}$

The various strings that can be obtained from Σ are

0, 1, 00, 01, 10, 11, 010, 101, ______

Length of a String | s |

Length of a string is the total numbers of symbols in the string.

Example :

Let $s = 001$ then, length of the string, $|s| = 3$

Empty String (∈): Empty string is a string of length zero.

1.2.1 – Operations on Strings

Concatenation

If x and y are two strings, then the concatenation of x and y denoted by $x \cdot y$ is the string obtained by writing the symbols of string x followed by the symbols of string y.

Example :

Let x = computer and y = science then,

$x \cdot y$ = computer science

Note

(i) The length of $x \cdot y$ is equal to the length of x and the length of y. i.e., $|x \cdot y| = |x| + |y|$

(ii) The concatenation of an empty string with any string is that string itself. i.e., $S \cdot \in \; = \; \in \cdot S = S$

Prefix of S: Prefix of S is any number of leading symbols of S.

Example :

Let $S = abc$ then prefix of $S = \in, a, ab, abc$

Suffix of S: Suffix of S is any number of trailing symbols of S.

Example :

Let $S = a\,b\,c$ then suffix of $S = \in, c, bc, abc$

Language: Language is a set of strings formed from some specific alphabet, Σ. The language, L is simply a subset of Σ^*, where Σ^* denotes the set of all strings over Σ.

Example :

Let $\Sigma = \{0, 1\}$ then, the language, $L = \{0, 01, 0011, 0111\}$

Where L consists of all the strings starting with zero.

Example :

Let $\Sigma = \{a, b\}$ then the language, $L = \{a^m b^m \mid m > 0\}$

Where L consists of all the strings beginning with one or more a's followed by same number of b's.

i.e., $L = \{\in, ab, aabb, aaabbb, \_\_\_\_\_\}$

Note

(i) A language containing empty string, $\in$ is denoted by $\{\in\}$.

(ii) An empty language is denoted by $\emptyset$ and do not have any members.

(iii) $\emptyset$ and $\{\in\}$ are not the same.

1.2.2 Operations of Languages

Concatenation (L.M)

If L and M are languages, then the concatenation of L and M denoted by L.M is the language consisting of all the strings xy which can be formed by selecting a string of x from L and a string of y from M and concatenating them in that order.

i.e., $L.M = \{xy \mid x \text{ is in } L \text{ and } y \text{ is in } M\}$

Example :

Let $L = \{0, 01, 110\}$ and $M = \{10, 110\}$ then $L.M = \{010, 0110, 01110, 11010, 110110\}$

Union ($L \cup M$): If L and M are languages, then Union of L and M, denoted by $L \cup$ M is the language consisting of the string x from L or the string x from M.

i.e., : $L \cup M = \{ x \mid x \text{ is in } L \text{ or } x \text{ is in } M\}$

Example :

Let $L = \{0, 01, 110\}$ and $M = \{10, 110\}$ then $L \cup M = \{0, 01, 10, 110\}$

Kleen Closure (L^*): Kleen closure of the language L, is the unary operation, L^* of the language L. That is zero or more concatenations of L.

$$L^* = \bigcup_{i=0}^{n} L^i = L^0 \cup L^1 \cup L^2 \_\_\_\_$$

Example :

Let, $L = \{0, 1\}$ then $L^* = \{\in, 0, 1, 00, 01, 11, 10, 000, \_\_\_\_\_\}$

Positive Closure (L^+): Positive closure, L^+ is the same as L^* but without empty string, $\in$. That is one or more concatenations of L.

$$L^+ = \bigcup_{i=1}^{n} L^i = L^1 \cup L^2 \cup L^3 \cup \_\_\_\_$$

Example :

Let $L = \{0, 1\}$ then $L^+ = \{0, 1, 00, 01, 11, 10, 000, \_\_\_\_\_\}$

Note $L^* = \{\in\} \cup L^+$

1.3 Finite Automata (FA)

Finite automata is a mathematical model which is used to study the abstract machines or abstract computing devices with the inputs chosen from Σ. Where Σ is the set of alphabets using which any string can be obtained. On reading the string, the machine may accept the string or reject the string.

Note

An abstract machine is a conceptual or theoretical model of a computer hardware or software system which really does not exist.

Finite automata is an abstract model of a digital computer which has the following three components

- Input tape • Control Unit and • Output
- **Input Tape:** The input tape is divided into cells each of which can hold one symbol. The string to be processed is stored in these cells.
- **Control Unit:** Control unit has finite set of states. It has one start state and one or more final states. The state of the machine may change based on the current input symbol.
- **Output:** Output may be accept or reject. When end of the input is encountered the finite automata accepts the input string if it is a final state/accepting state, otherwise it rejects the input string.

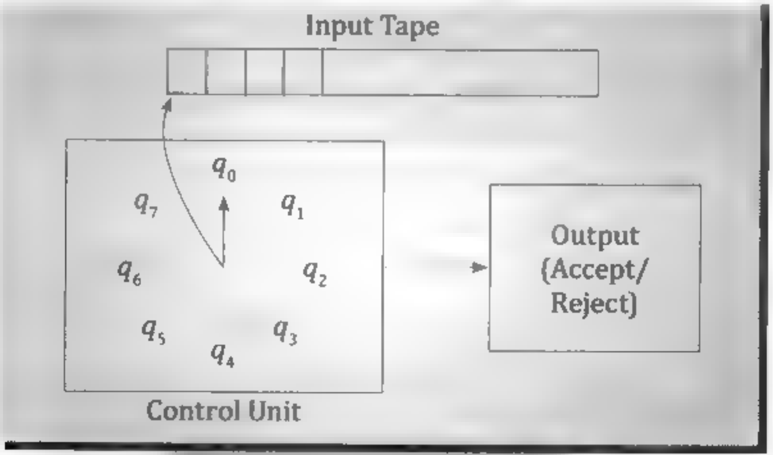


Fig. 1.1: Components of FA

1.3.1 - Mathematical Representation of Finite Automata

Finite automata is a five tuple, $(Q, \Sigma, \delta, q_0, F)$

Where Q is a non-empty finite set of states.

Σ is non-empty finite set of input symbols.

$q_0 \in Q$, is the start state.

$F \subseteq Q$, is a set of accepting/final states.

$\delta : Q \times \Sigma \longrightarrow Q$, is a transition function.

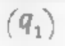
For any symbol $a \in \Sigma$ and states $q_0, q_1, \in Q$, $\delta(q_0, a) = q_1$ implies that the final automata moves from state q_0 to q_1 on receiving the input symbol a.

1.3.2 - Transition Diagram

Finite automata is represented using a transition diagram/labeled directed graph, where the nodes represents the states and the labeled edges represents the transition function

Notations used to represent a FA

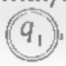
(i) Nodes are called states and each state is represented by a circle.

 , q_1 is a state.

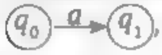
(ii) The start state is represented using a circle with an arrow which is not originating from any node.

, q_0 is a start state

(iii) Final/Accepting state is denoted by double circle.

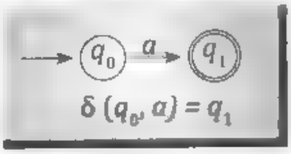
, q_1 is a final/accepting state

(iv) The transitions are marked by arrow or directed line labelled with the input symbol, which starts from the current state and ends in the next state.

, $\delta(q_0, a) = q_1$

Example :

Transition Diagram



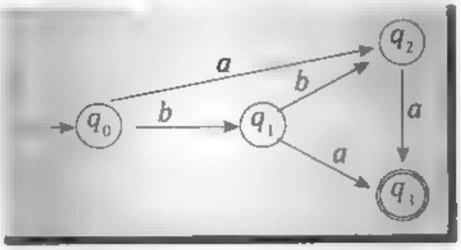
Transition Table

δ	a
$\rightarrow q_0$	q_1
$* q_1$	$\emptyset$

1.3.3 - Transition Table

The transitions of a FA can also be represented using a transition table, in which there is a row for each state and a column for each input symbol. In the transition table the start state is marked with an arrow and the final/accepting state with a star.

Example :



FA is defined as

$M = (Q, \Sigma, \delta, q_0, F)$

Where, $Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ	a	b
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_2
q_2	q_3	Ø
$* q_3$	Ø	Ø

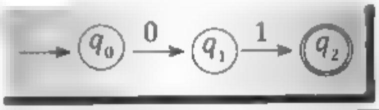
1.3.4 Language Accepted by a FA

Any string using which we can traverse from the start state to the final state is said to be accepted by a finite automata.

$L(M) = \{x | \delta (q_0, x) \text{ is in } F\}$

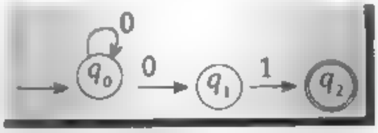
Example 1 :

Finite automata which accepts an input string 01.



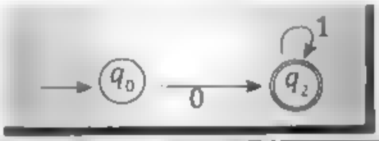
Example 2 :

Finite automata which accepts an infinite number of 0's followed by 01.



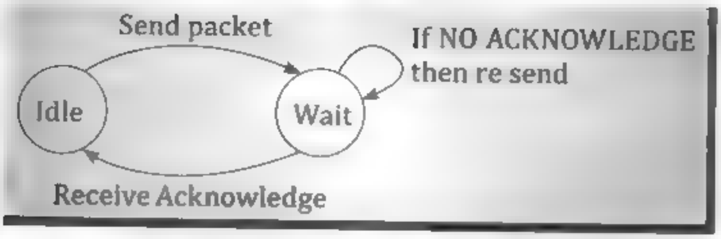
Example 3 :

Finite automata which accepts a 0 followed by any number of 1's.



1.4. Applications of FA

- **Language Processing:** The applications of finite automate are found in language processing such as string processing, parsing, morphological representations, information retrieval, tagging, stemming, image compressions etc.,
- **Compiler constructions**: FA is used in the design of first phase of compiler (ie.,) Lexical Analyser, which breaks the input text into tokens such as identifiers, keywords, punctuations etc.,
- **Computer Networks:** FA is used in the design of communication protocols. For example, for a communication link.



- **Video Games:** Games like Warcraft III, take advantage of complex FA to control artificial intelligence. A bot is a computer generated character in a video games. Chat dialogues where the user is prompted with choices can also be run using FA.
- **Design of Digital Circuits:** FA is used during designing and checking the behaviour of digital circuits using software. FA is useful in the design of automatic traffic signals and circuit verification.
- **Biomedical Problem Solving:** Fourier transform Nuclear Magnetic Resonance (NMR) imaging, X-ray tomography, X-ray diffraction, differential scanning calorimetry and mass spectrometry are some of the techniques which employ FA.
- Cellular automata are used for making pretty pictures and animations.

1.5 - Types of FA

The different types of finite automata are

- Deterministic finite Automata (DFA)
- Non-Deterministic Finite Automata (NFA)
- Non-Deterministic Finite Automata with ∈ – moves (∈ – NFA)

1.6 - Deterministic Finite Automata (DFA)

DFA is a finite automata which can have only one transition from a state on an input symbol. That is for each state '*q*' and input symbol '*a*', there is at most one edge labeled '*a*' leaving '*q*'. A DFA returns exactly one state after reading an input symbol.

1.6.1 - Mathematical Representation of a DFA

Definition: DFA is a five tuple $(Q, \Sigma, \delta, q_0, F)$

Where Q is a non-empty finite set of states

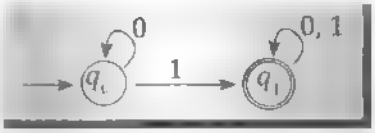
Σ is a non-empty finite set of input symbols

$q_0 \subseteq Q$, is the start state

$F \subseteq Q$, is a set of accepting/final states

$\delta : Q \times \Sigma \longrightarrow Q$, is a transition function given by the mapping function $Q \times \Sigma \longrightarrow Q$

Example :



Where

DFA is defined as $D = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_1\}$

δ	0	1
$\rightarrow q_0$	q_0	q_1
$^*q_1$	q_1	q_1

1.6.2 - Extended Transition Function of DFA (δ^*)

The extended transition function, δ^* describes what happens to a state of machine when the input is a string.

Let $D = (Q, \sum, \delta, q_0, F)$ be a DFA

The extended transition function, $\delta^* : Q \; X \sum^* \longrightarrow Q$ is defined recursively as follows.

Basis: $\delta^* (q, \epsilon) = q$

That is the DFA stays in the same state 'q', when it reads an empty string ϵ.

Let $w = xa,$

Where 'a' is the last symbol of w and 'x' is the remaining string of w.

Let 'q' be the current state and w be the string to be processed. After consuming the string w, let the state of the machine be 'p'.

Then, $\delta^* (q, w) = \delta^* (q, x\,a)$

$= \delta^* (\delta^*(q, x), a) = p$

Properties of Extended Transition Function

1. $\delta^* (q, \epsilon) = q$
2. $\delta^* (q, w) = \delta^* (q, x\,a) = \delta\, (\delta^* (q, x), a)$, where $w = x\,a$
3. $\delta^* (q, w) = \delta^* (q, a\,x) = \delta^* (\delta\, (q, a), x)$, where $w = a\,x$

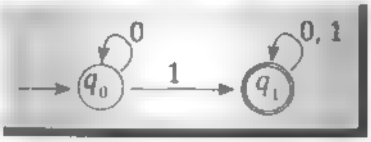
1.6.3 - Language Accepted by a DFA

Let $D = (Q, \sum, \delta, q_0, F)$ be a DFA. A string w is accepted by the DFA D if it takes the initial state q_0 to final state ie., $\delta^* (q_0, w)$ is in F.

$$L(M) = \{w | w \in \sum^* \text{ and } \delta^* (q_0, w) \text{ is in F}\}$$

Problem 1

Show that the string 0011 is accepted by the following DFA



Solution

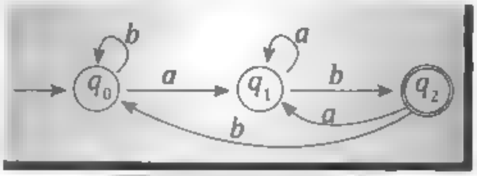
To show the string 0011 is accepted by the DFA, we use the following moves:

$$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1$$

Since we started from the start state q_0 and successively traversed the states with the input string and reached the final state, the string 0011 is accepted by the DFA.

Problem 2

Check whether the strings *a a ba b* and *b a b a* is accepted by the following DFA.



Solution

(i) **For the string:** *a a b a b*

Moves are as follows: $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_1 \xrightarrow{b} q_2$

Since we encountered the end of the input and we are in the final state, the string is accepted by the DFA.

(ii) **For the string:** *baba*

Moves are as follows: $q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_1$

Since we encountered the end of the input and q_1 is not the final state, the string is rejected by the DFA.

1.6.4 - DFA Design

Different types of problems for which we can construct a DFA are

- Pattern recognition problems
- Divisible by K problems
- Modulo - K - counter problems.

1.6.4.1 Pattern Recognition Problems

The general steps to be followed while designing the DFA are

1. Identify the minimum string
2. Identify the alphabets
3. Construct a skeleton of DFA
4. Identify other transitions which are not defined in step 3.
5. Construct the DFA using transitions in step 3 and step 4.

PROBLEM 4

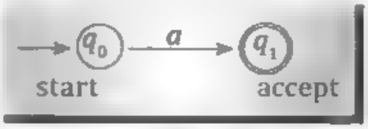
Draw a DFA to accept strings of a's having atleast one *a*.

SOLUTION

Step 1: Identify the minimum string = *a*

Step 2: Identify the alphabets, $\Sigma = \{a\}$

Step 3: Construct the skeleton DFA.

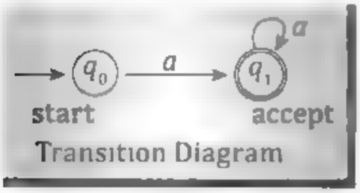


Step 4: Identify the transitions not defined in Step 3.

$\delta(q_1, a) = ?$

Since after one a, any number of *a*'s can be accepted, the transition is $\delta(q_1, a) = q_1$

Step 5: Construct the DFA.



Transition Diagram

The DFA is defined as

$D = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{q_0\, q_1\}$

$\Sigma = \{a\}$

$q_0 = \{q_0\}$

$F = \{q_1\}$

δ	a
$\rightarrow q_0$	q_1
$*q_1$	q_1

PROBLEM 4

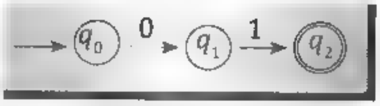
Design a DFA for strings that always ends with 01 where $\Sigma = \{0, 1\}$

SOLUTION

Step 1: minimum string = 01

Step 2: $\Sigma = \{0, 1\}$

Step 3: Skeleton DFA



Step 4: Identify the transitions not defined in Step 3

δ	0	1
$\rightarrow q_0$	q_1	?
q_1	?	q_2
* q_2	?	?

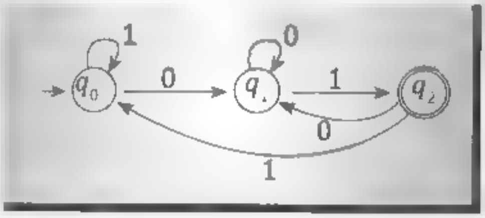
$\delta(q_0, 1) = q_0$ since the string can begin with any number of 1's

$\delta(q_1, 0) = q_1$ since the string can have any number of 0's as the substring

$\delta(q_2, 0) = q_1$ since the string has to end with 01.

$\delta(q_2, 1) = q_0$ since the string has to end with 01

Step 5: Construct the DFA



The DFA is defined as

$$D = \{Q, \Sigma, \delta_1, q_0, F\}$$

where $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_2\}$

δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
* q_2	q_1	q_0

PROBLEM 5

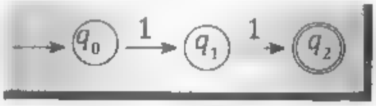
Design a DFA to accept the set of strings of 0 and 1 ending with 11.

SOLUTION

Step 1: minimum string = 11

Step 2: $\Sigma = \{0, 1\}$

Step 3: Skeleton DFA



Step 4: Identify the transitions not defined in Step 3

δ	0	1
$\rightarrow q_0$	?	q_1
q_1	?	q_2
* q_2	?	?

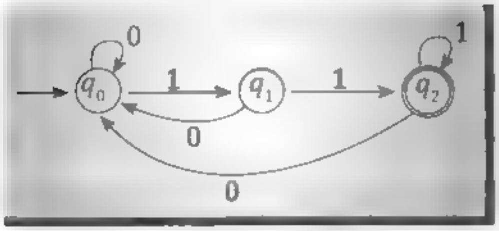
$\delta(q_0, 0) = q_0$ The string can begin with any number of 0's

$\delta(q_1, 0) = q_0$ The string can have any number of 0's as substring but has to end with 11.

$\delta(q_2, 0) = q_0$ The string has to end with 11

$\delta(q_2, 1) = q_2$ The string can have more than two 1's at the end.

Step 5: construct the DFA



The DFA is defined as

D = {Q, ∑, δ, q_0, F}

where $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_2\}$

δ	0	1
→ q_0	q_0	q_1
q_1	q_0	q_2
* q_2	q_0	q_2

Problem 6

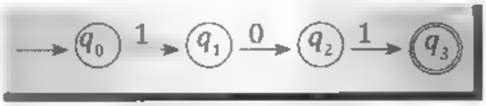
Draw a DFA to accept strings of 0's and 1's ending with 101.

Solution

Step 1: Minimum string = 101

Step 2: ∑ = { 0, 1}

Step 3: Skeleton DFA



$$\begin{tikzcd} \arrow[r] & q_0 \arrow[r, "1"] & q_1 \arrow[r, "0"] & q_2 \arrow[r, "1"] & q_3 \end{tikzcd}$$

Step 4: Identify the other undefined transitions.

δ	0	1
→ q_0	?	q_1
q_1	q_2	?
q_2	?	q_3
* q_3	?	?

δ (q_0, 0) = q_0 The string can begin with any number of 0's

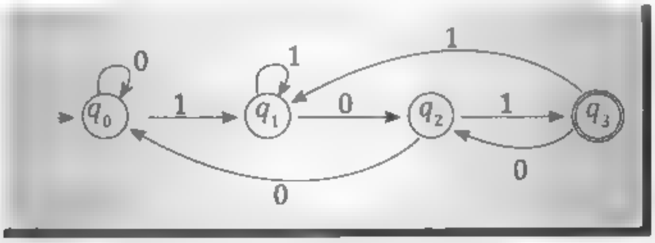
δ (q_1, 0) = q_0 The string has to end with 101.

δ (q_2, 0) = q_0 The string can have any number of 0's as the substring.

δ (q_3, 1) = q_2 Since δ (q_2, 1) = q_3 and the sequence 101 is accepted.

δ (q_3, 1) = q_1 The string has to end with 101.

Step 5: construct the DFA



∴ The DFA is defined as

D = {Q, ∑, δ, q_0, F}

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$q_0 = \{q_0\}$
$F = \{q_3\}$

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_3
$* \; q_3$	q_2	q_1

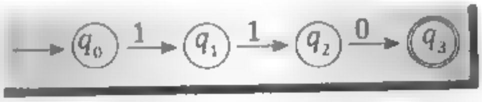
Problem 7

Draw a DFA to accept strings which end with 110 where $\Sigma = \{0, 1\}$

Solution

Step 1: Minimum string = 110

Step 2: $\Sigma = \{0, 1\}$

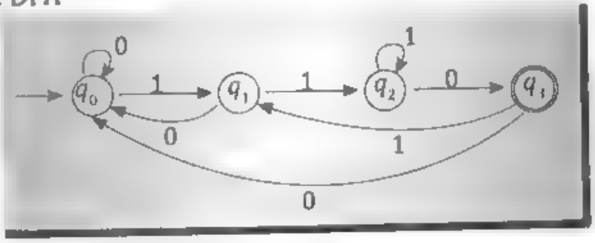
Step 3: Skeleton DFA 

Step 4: Identify the other undefined transitions.

δ	0	1
$\rightarrow q_0$	?	q_1
q_1	?	q_2
q_2	q_3	?
$* \; q_3$	?	?

$\delta(q_0, 0) = q_0$ The string can begin with any number of 0's
$\delta(q_1, 0) = q_0$ The string can have any number of 0's as substring.
$\delta(q_2, 1) = q_2$ The sequence 110 is accepted
$\delta(q_3, 1) = q_0$ The string has to end with 110
$\delta(q_3, 1) = q_1$ The sequence 110 is accepted

Step 5: Construct the DFA



∴ The DFA is defined as

$\mathbf{D} = \{Q, \Sigma, \delta, q_0, F\}$
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$q_0 = \{q_0\}$
$F = \{q_3\}$

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_0	q_2
q_2	q_3	q_2
$* \; q_3$	q_0	q_1

PROBLEM 8

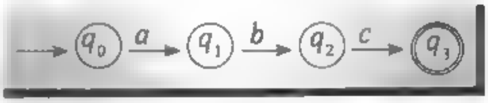
Draw a DFA to accept strings of *a*'s and *b*'s ending with abc.

SOLUTION

Step 1: minimum string abc

Step 2: $\Sigma = \{a, b, c\}$

Step 3: Skeleton DFA



Step 4: Identify the other undefined transitions.

δ	a	b	c
$\rightarrow q_0$	q_1	?	?
q_1	?	q_2	?
q_2	?	?	q_3
* q_3	?	?	?

$\delta(q_0, b) = q_0$ string can begin with any number of *b*'s

$\delta(q_0, c) = q_0$ string can begin with any number of *c*'s

$\delta(q, a) = q_1$ string can have any number of *a*'s as substring but should end with *abc*.

$\delta(q_1, c) = q_0$ string has to end with *abc*.

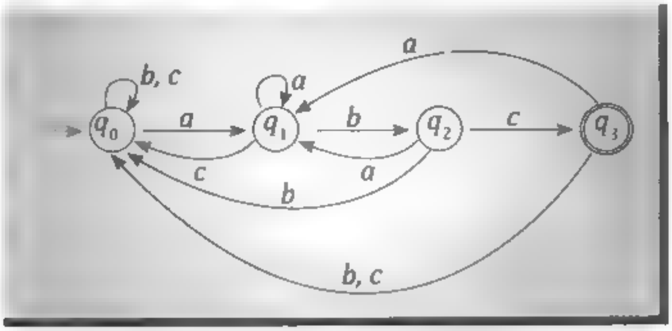
$\delta(q_2, a) = q_0$ string has to end with *abc*.

$\delta(q_2, b) = q_0$ string can have any number of *b* as substring.

$\delta(q_3, a) = q_1$ string has to end with abc

$\left.\begin{array}{l}\delta(q_3, b) = q_0 \\ \delta(q_3, c) = q_0\end{array}\right\}$ string can have any number of *b* or *c* or combination of *b* and *c* as substring.

Step 5: construct the DFA



$\therefore$ **DFA D is defined as D = {*Q*, Σ, δ, q_0, *F* }**

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b, c\}$
$q_0 = \{q_0\}$
$F = \{q_3\}$

δ	a	b	c
$\rightarrow q_0$	q_1	q_0	q_0
q_1	q_1	q_2	q_0
q_2	q_1	q_0	q_3
* q_3	q_1	q_0	q_0

Problem 9

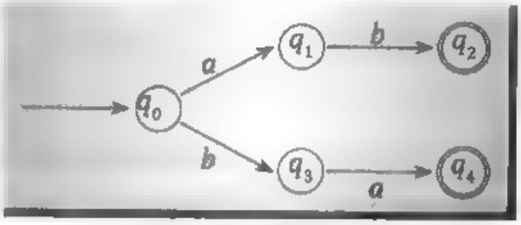
Draw a DFA to accept strings of *a*'s an *b*'s ending with *ab* or *ba*.

Solution

Step 1: Minimum string *ab* or *ba*

Step 2: $\Sigma = \{ a, b \}$

Step 3: Skeleton DFA



Step 4: Skeleton DFA

δ	a	b
$\rightarrow q_0$	q_1	q_3
q_1	?	q_2
$*q_2$	?	?
q_3	q_4	?
$*q_4$	?	?

$\delta(q_1, a) = q_1$ string can have any number of a as substring but end with *ab*.

$\delta(q_2, a) = q_4$ string can end with *ba*

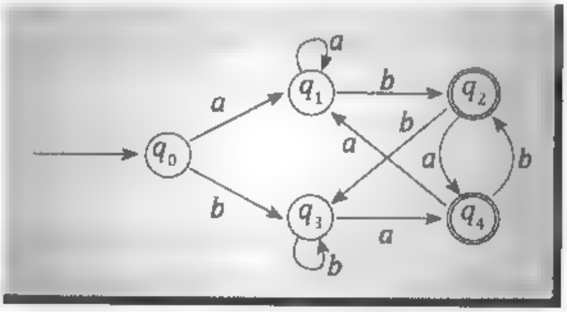
$\delta(q_2, b) = q_3$ The sequence abb is not accepted, but abb followed by a is accepted as it ends with ba

$\delta(q_3, b) = q_3$ string can have any number of *b* as substring

$\delta(q_4, a) = q_1$ The sequence baa is not accepted but baa followed by *b* is accepted as it ends with *ab*.

$\delta(q_4, b) = q_2$ string can end with ab

Step 5: Construct the DFA



$\therefore$ DFA is defined as

$\mathbf{D = \{Q, \Sigma, \delta, q_0, F\}}$

$Q = \{ q_0, q_1, q_2, q_3, q_4 \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ q_0 \}$

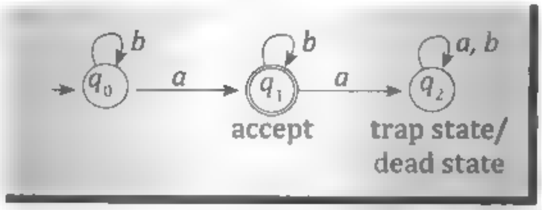
$F = \{ q_2, q_4 \}$

δ	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_1	q_2
$*q_2$	q_4	q_3
q_3	q_4	q_3
$*q_4$	q_1	q_2

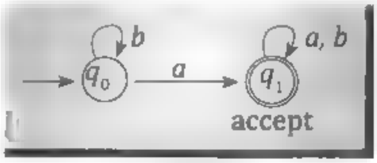
Trap State: A state for which there exists transitions to itself for all the input symbols chosen from Σ is called a trap state.

Dead State: A trap state can be a non final state or a final state. A non final trap state is called dead state.

Example :



A final state, which is a trap state.



Problem 10

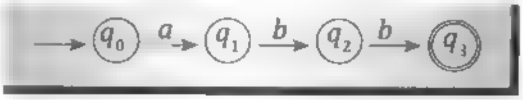
Draw a DFA to accept string of *a*'s and *b*'s ending with abb, and also show that the strings bbabb and aaabb are accepted by the DFA.

Solution

Step 1: minimum string abb

Step 2: $\Sigma = \{ a, b \}$,

Step 3: Skeleton DFA



Step 4: Identify other transitions

δ	a	b
$\rightarrow q_0$	q_1	?
q_1	?	q_2
$*q_2$	?	q_3
q_3	?	?

$\delta(q_0, b) = q_0$ The string can begin with any number of *b*'s

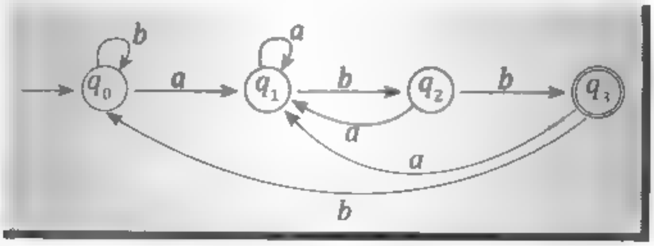
$\delta(q_1, a) = q_1$ The string can have any number of *a*'s as substring but has to end with abb

$\delta(q_2, a) = q_1$ The sequence aba must follow bb so as to end with abb

$\delta(q_3, a) = q_1$ since the string has to end with abb

$\delta(q_3, b) = q_0$ The string can have any number o *b*'s as substring.

Step 5: Construct the DFA



. DFA is defined as

$D = \{Q, \Sigma, \delta, q_0, F\}$
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$q_0 = \{q_0\}$
$F = \{q_3\}$

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
$*q_3$	q_1	q_0

Checking the acceptability of the string *bbabb.*

The moves are as follows

$$q_0 \xrightarrow{b} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{b} q_3$$

Since we where able to successfully traverse the string bbabb from the starting state, q_0 to the final state q_3, the string is accepted by the DFA.

Checking the acceptability of the string *aaabb*

The moves are as follows

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{b} q_3$$

Since we could traverse the string aaabb from the starting state q_0 to the final state q_3, the string is accepted by the DFA.

Problem 11

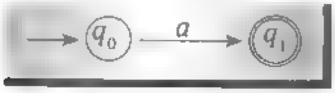
Draw a DFA to accept strings of *a*'s and *b*'s having exactly one a.

Solution

Step 1: Identify minimum string : a

Step 2: $\Sigma = \{a, b\}$

Step 3: Skeleton DFA



Step 4: Identify other transitions.

δ	a	b
$\rightarrow q_0$	q_1	?
$*q_1$	?	?

$\delta(q_0, b) = q_0$ string can begin with any number of a's

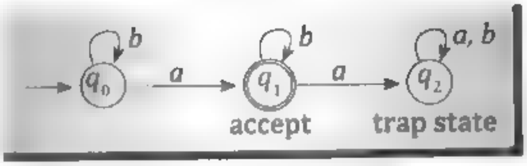
$\delta(q_1, a) = q_2$ q_2 is a trap state because almost one a is allowed in the string.

$\delta(q_1, b) = q_1$ string can end with any number of b's

$\delta(q_2, a) = q_2$, q_2 is a trap state

$\delta(q_2, b) = q_2$, q_2 is a trap state

Step 5: Construct the DFA



∴ DFA is defined as

D = {Q, Σ, δ, q_0, F}

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ	a	b
$\rightarrow q_0$	q_1	q_0
$*q_1$	q_2	q_1
q_2	q_2	q_2

Problem 12

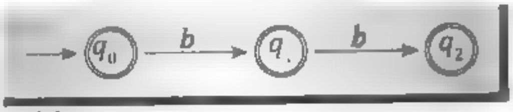
Obtain a DFA to accept strings of *a*'s and *b*'s with atmost two consecutive *b*'s.

Solution

Step 1: Minimum string : bb

Step 2: $\Sigma = \{a, b\}$

Step 3: Skeleton DFA



Step 4: Identify other transitions

δ	a	b
$\rightarrow *q_0$	?	q_1
$*q_1$	?	q_2
$*q_2$	?	?

$\delta(q_0, a) = q_0$ string beginning with any number of *a*'s is accepted

$\delta(q_1, a) = q_0$ string with any number of *a*'s as substring is accepted

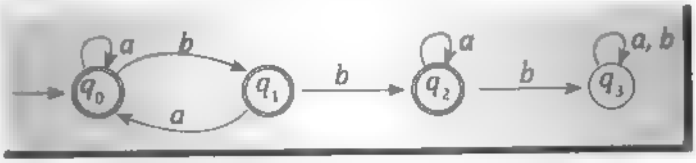
$\delta(q_2, a) = q_2$ any number of *a*'s can occur at any position in the string

$\delta(q_2, b) = q_3$, q_3 is trap state as atmost two consecutive *b*'s are allowed

$\delta(q_3, a) = q_3$, q_3 is a dead state

$\delta(q_3, b) = q_3$, q_3 is a dead state.

Step 5: Construct the DFA



∴ DFA is defined as

$D = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_0, q_1, q_2\}$

δ	a	b
$\rightarrow *q_0$	q_0	q_1
$*q_1$	q_0	q_2
$*q_2$	q_2	q_3
q_3	q_3	q_3

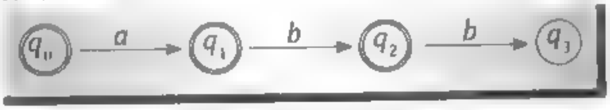
Problem 14

Draw a DFA to accept strings of *a*'s and *b*'s which do not end with abb.

Solution

Step 1: $\Sigma = \{a, b\}$

Step 2: The skeleton DFA will be



Since the DFA do not end with abb q_1 cannot be a final state. q_0, q_1, q_2 are final states as the string can have zero or more number of *a*'s and *b*'s as the substring.

Step 3: Identify the other transitions.

δ	a	b
$\rightarrow *q_0$	q_0	?
$*q_1$	?	q_2
$*q_2$	?	q_3
q_3	?	?

$\delta(q_0, b) = q_0$ the string can begin with any number of *b*'s.

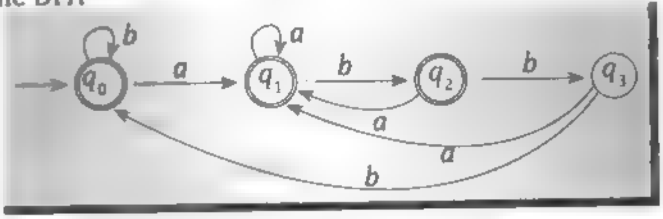
$\delta(q_1, a) = q_1$ string can have any number of *a*'s as substring

$\delta(q_2, a) = q_1$ string can end with *a* or *ab*

$\delta(q_3, a) = q_1$ string should not end with abb.

$\delta(q_3, b) = q_0$, q_0 string should not end with abb

Step 4: construct the DFA



∴ DFA is defined as

$D = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_0, q_1, q_2\}$

δ	a	b
$\rightarrow *q_0$	q_1	q_0
$*q_1$	q_1	q_2
$*q_2$	q_1	q_3
q_3	q_1	q_0

Problem 15

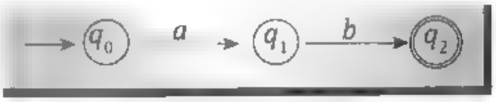
Obtain a DFA to accept strings of *a*'s and *b*'s starting with the string *ab*.

Solution

Step 1: Minimum string : ab

Step 2: $\Sigma = \{ a, b \}$

Step 3: Skeleton DFA



Step 4: Identify other transitions.

δ	a	b
$\rightarrow q_0$	q_1	?
q_1	?	q_2
$*q_2$	?	?

$\delta(q_0, b) = q_3$ q_3 is a dead state as the string has to begin with *ab*.

$\delta(q_1, a) = q_3$ string has to begin with *ab*

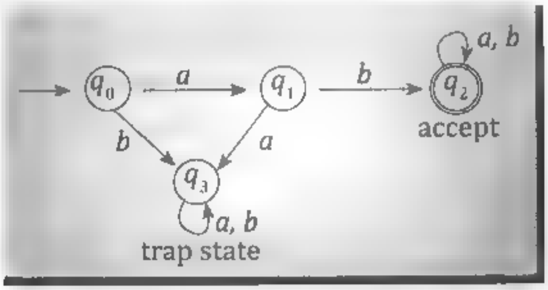
$\delta(q_2, a) = q_2$ string can have any number of *a*'s as substring

$\delta(q_2, b) = q_2$ string can have any number of b as substring

$\delta(q_3, a) = q_3$, q_3 is a dead state

$\delta(q_3, b) = q_3$, q_3 is a dead state

Step 5: construct the DFA



. DFA is defined as

$D = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{ q_0, q_1, q_2, q_3 \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ q_0 \}$

$F = \{ q_3 \}$

δ is shown by the transition table

δ	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_3	q_2
$*q_2$	q_2	q_2
q_3	q_3	q_3

PROBLEM 16

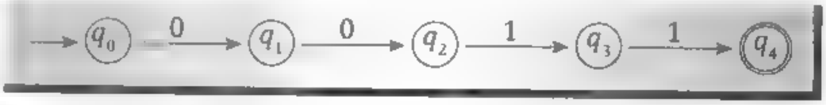
Obtain a DFA to accept strings of 0's and 1's starting with atleast two 0's and ending with atleast two 1's

SOLUTION

Step 1: Minimum string 0011

Step 2: $\Sigma = \{0, 1\}$

Step 3: Skeleton DFA



Step 4: Identify other transitions

δ	0	1
$\rightarrow q_0$	q_1	?
q_1	q_2	?
q_2	?	q_3
q_3	?	q_4
$*q_4$	?	?

$\delta(q_0, 1) = q_5$ string has to begin with 00. Beginning the string with 1 leads to dead state.

$\delta(q_1, 1) = q_5$ string has to begin with 00. Beginning with 01 is not accepted

$\delta(q_2, 0) = q_2$ string can have any number of 0's as substring

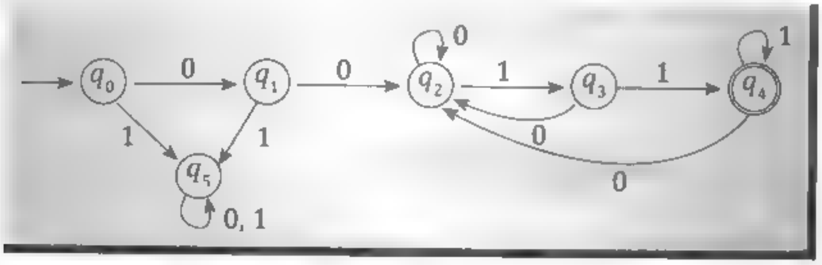
$\delta(q_3, 0) = q_2$ string has to begin with 11

$\delta(q_4, 0) = q_2$ string can have any number of 0's as substring and has to end with 11.

$\delta(q_4, 1) = q_4$ ending with more than two 1's are allowed $\delta(q_5, 0) = q_5$, q_5 is a dead state

$\delta(q_5, 1) = q_5$, q_5 is a dead state

Step 5: Construct the DFA



$\therefore$ DFA is defined as

$D = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_4\}$

δ	0	1
$\rightarrow q_0$	q_1	q_5
q_1	q_2	q_5
q_2	q_2	q_3
q_3	q_2	q_4
$*q_4$	q_2	q_4
q_5	q_5	q_5

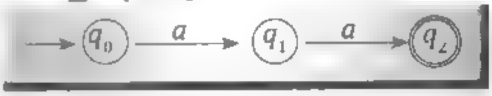
PROBLEM 17

Draw a DFA that accepts strings of *a*'s and *b*'s having a substring *aa*.

SOLUTION

Step 1: Minimum string = *aa*

Step 2: Identify alphabets $\sum = \{ a, b \}$

Step 3 : Skeleton DFA 

$$\begin{tikzcd} \arrow[r] & q_0 \arrow[r, "a"] & q_1 \arrow[r, "a"] & q_2 \end{tikzcd}$$

Step 4: Identify other transitions

δ	a	b
$\rightarrow q_0$	q_1	?
q_1	q_2	?
q_2	?	?

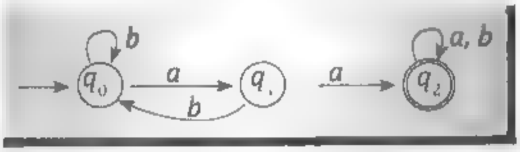
$\delta(q_0, b) = q_0$ string can begin with any number of b's

$\delta(q_1, b) = q_0$ string should have aa as substring

$\delta(q_2, a) = q_2$ string can end with any number of a's

$\delta(q_2, b) = q_2$ string can end with any number of b's

Step (v) construct the DFA



$$\begin{tikzcd} \arrow[r] & q_0 \arrow[loop above, "b"] \arrow[r, "a"] & q_1 \arrow[l, bend left, "b"] \arrow[r, "a"] & q_2 \arrow[loop above, "{a, b}"] \end{tikzcd}$$

∴ DFA is defined as

D = {$Q, \Sigma, \delta, q_0, F$}

$Q = \{ q_0, q_1, q_2 \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ q_0 \}$

$F = \{ q_2 \}$

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
$*q_2$	q_2	q_2

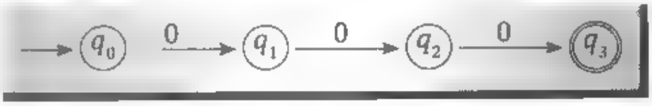
PROBLEM 18

Construct a DFA that accepts the set of all strings with 3 consecutive 0's, where $\Sigma = \{ 0, 1 \}$

SOLUTION

Step 1: Identify minimum string = 000

Step 2: $\Sigma = \{ 0, 1 \}$

Step 3: Skeleton DFA 

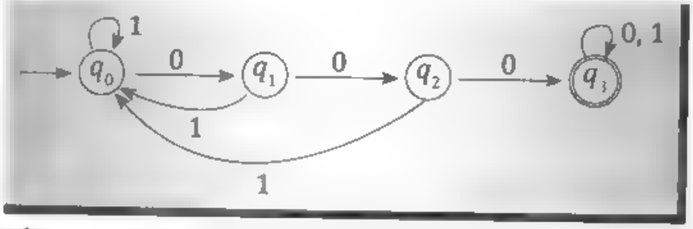
$$\begin{tikzcd} \arrow[r] & q_0 \arrow[r, "0"] & q_1 \arrow[r, "0"] & q_2 \arrow[r, "0"] & q_3 \end{tikzcd}$$

Step 4: Identify other transitions

δ	0	1
$\rightarrow q_0$	q_1	?
q_1	q_2	?
q_2	q_3	?
$*q_3$	?	?

$\delta(q_0, 1) = q_0$ string can begin with any number of 1's
$\delta(q_1, 1) = q_0$ string should have 3 consecutive 0's
$\delta(q_2, 1) = q_0$ string should have 3 consecutive 0's
$\delta(q_3, 0) = q_3$ string can end with any number of 0's
$\delta(q_3, 1) = q_3$ string can have any number of is at the end

Step 5: Construct the DFA



∴ DFA is defined as

D = {$Q, \Sigma, \delta, q_0, F$}
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$q_0 = \{q_0\}$
$F = \{q_3\}$

δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
$*q_3$	q_3	q_3

Problem 19

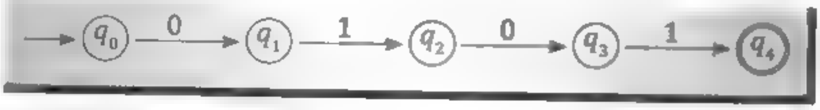
Draw a DFA over alphabet $\Sigma = \{0, 1\}$, that contain set of strings of 0's and 1's which contain substring 0101.

Solution

Step 1: Minimum string = 0101

Step 2: $\Sigma = \{0, 1\}$

Step 3: Skeleton DFA

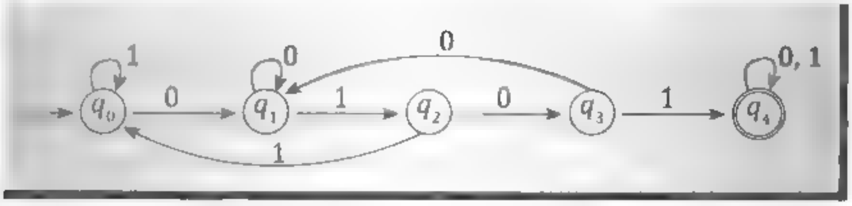


Step 4: Identify other transitions

δ	0	1
$\rightarrow q_0$	q_1	?
q_1	?	q_2
q_2	q_3	?
q_3	?	q_4
$*q_4$	?	?

$\delta(q_0, 1) = q_0$ string can begin with any number of 0's
$\delta(q_1, 0) = q_1$ string can have any number of 0's as substring
$\delta(q_2, 1) = q_0$ string can have any number of 1's as substring
$\delta(q_3, 0) = q_1$ string can have any number of 0's as substring
$\delta(q_4, 0) = q_4$ string can end with any number of 0's
$\delta(q_4, 1) = q_4$ string can end with any number of 1's

Step 5: Construct the DFA



∴ DFA is defined as

D = {$Q, \Sigma, \delta, q_0, F$}

$Q = \{ q_0, q_1, q_2, q_3, q_4 \}$

$\Sigma = \{ 0, 1 \}$

$q_0 = \{ q_0 \}$

$F = \{ q_4 \}$

δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_3	q_0
q_3	q_1	q_4
$*q_4$	q_4	q_4

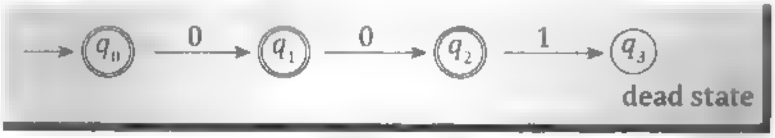
PROBLEM 20

Design a DFA over alphabet $\Sigma = \{ 0, 1 \}$ that contains set of strings of 0's and 1's except those containing substring 001.

SOLUTION

Step 1: $\Sigma = \{ 0, 1 \}$

Step 2: The skeleton DFA will be



Since the DFA do not accept substring 001, q_3 is a dead state.

q_0, q_1, q_2 are final states as the string can have zero or more number of 0's and 1's.

Step 3: Identify other transitions

δ	0	1
$\rightarrow q_0$	q_1	?
$*q_1$	q_2	?
$*q_2$	?	q_3
q_3	?	?

$\delta(q_0, 1) = q_0$ string can begin with any number of 1's

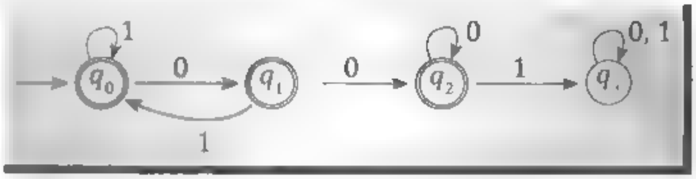
$\delta(q_1, 1) = q_0$ string can have any number of 1's as substring

$\delta(q_2, 0) = q_2$ string can end with any number of 0's

$\delta(q_3, 0) = q_3$, q_3 is a dead state

$\delta(q_3, 1) = q_3$, q_3 is a dead state

Step 4: Construct the DFA



∴ DFA is defined as

D = {*Q*, Σ, δ, *q*$_0$, *F*}

Where

$Q = \{ q_0, q_1, q_2, q_3 \}$

$\Sigma = \{ 0, 1 \}$

$q_0 = \{ q_0 \}$

$F = \{ q_0, q_1, q_2 \}$

δ	0	1
$\rightarrow *q_0$	q_1	q_0
$*q_1$	q_2	q_0
$*q_2$	q_2	q_3
q_3	q_3	q_3

PROBLEM 21

Construct a DFA to accept the set of all strings not containing the substring 00 for Σ = { 0, 1 } and show the acceptability of the string, *w* = 101011

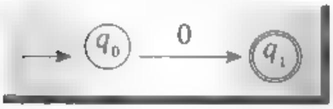
SOLUTION

Step 1: Minimum String = 0, 1

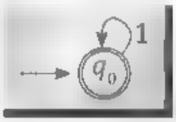
Step 2: Σ = { 0, 1 }

Step 3: Skeleton DFA

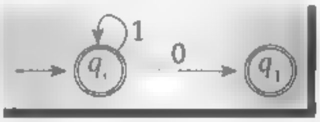
for 0



for 1



∴ the skelton DFA for 0 or 1 is



Step 4: Identify other transitions

δ	0	1
$\rightarrow q_0$	q_1	q_0
$*q_1$	?	?

$\delta(q_1, 0) = q_2$, q_2 is a new dead state since the DFA should not accept the substring 00.

$\delta(q\ , 1) = q_3$, q_1 is a new final state, since the DFA can accept the string 01.

Undefined transitions from q_2

$\delta(q_2, 0) = q_2$ since q_2 is a dead state

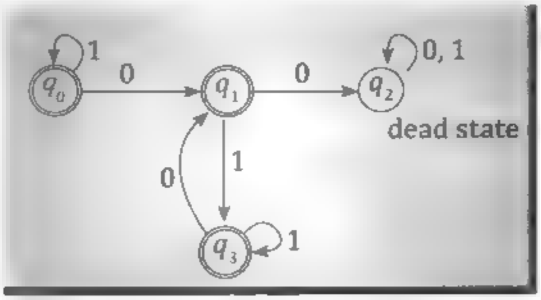
$\delta(q_2, 1) = q_2$

Undefined transitions from q_3

$\delta(q_3, 0) = q_1$ since it can accept the string 010.

$\delta(q_3, 1) = q_3$ since we can have any number of 1's at the end.

Step 5: Construct the DFA



∴ DFA is defined as **$D = \{Q, \Sigma, \delta, q_0, F\}$**

Where $Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_0, q_1, q_3\}$

δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_3
q_2	q_2	q_2
$*q_3$	q_1	q_3

Checking the acceptability of the string 101011.

Moves are as follows

$$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_3 \xrightarrow{0} q_1 \xrightarrow{1} q_3 \xrightarrow{1} q_3$$

Since q_3 is the final state, the string is accepted by the DFA.

1.6.4.2 Divisible by *k* Problems

In Divisible by *k* problems, the string should be divisible by *k*.

The steps to be followed to find DFA is shown below:

Step 1: Identify the radix *r*, input alphabet $\sum$, and the devisor *k*.

Step 2: Compute the possible remainders. These remainders represents the states of DFA

Step 3: Find the transitions using the relation $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$

r - radix of input. For binary r = 2.

i - remainder obtained after dividing by *k*.

d - digits. For binary d = {0, 1}

k - divisor

Step 4: Construct the DFA using the transitions obtained in step (iii)

PROBLEM 22

Construct a DFA which accepts only those strings representing zero modulo Five.

OR

Draw a DFA to accept binary strings divisible by 5.

SOLUTION

The strings representing zero modulo five are 0000, 0101, 1010, 1111, etc.

Step 1: Identify radix, input alphabets, and the divisor

$r = 2, d = \{0, 1\}\ k = 5$

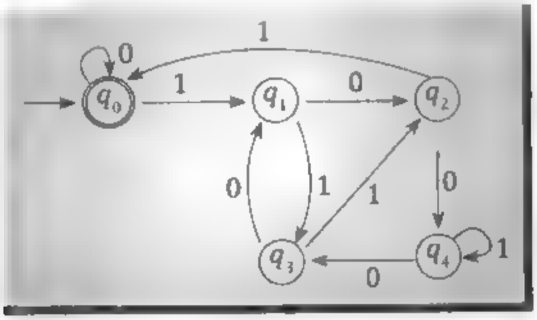
Step 2: Compute the possible remainders.

$i = 0, 1, 2, 3, 4$

Step 3: Compute the transitions using the relation $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$.

Remainder	d	$(2 * i + d) \bmod 5 = j$	$\delta(q_i, d) = q_j$
$i = 0$	0	$(2 * 0 + 0) \bmod 5 = 0$	$\delta(q_0, 0) = q_0$
	1	$(2 * 0 + 1) \bmod 5 = 1$	$\delta(q_0, 1) = q_1$
$i = 1$	0	$(2 * 1 + 0) \bmod 5 = 2$	$\delta(q_1, 0) = q_2$
	1	$(2 * 1 + 1) \bmod 5 = 3$	$\delta(q_1, 1) = q_3$
$i = 2$	0	$(2 * 2 + 0) \bmod 5 = 4$	$\delta(q_2, 0) = q_4$
	1	$(2 * 2 + 1) \bmod 5 = 0$	$\delta(q_2, 1) = q_0$
$i = 3$	0	$(2 * 3 + 0) \bmod 5 = 1$	$\delta(q_3, 0) = q_1$
	1	$(2 * 3 + 1) \bmod 5 = 2$	$\delta(q_3, 1) = q_2$
$i = 4$	0	$(2 * 4 + 0) \bmod 5 = 3$	$\delta(q_4, 0) = q_3$
	1	$(2 * 4 + 1) \bmod 5 = 4$	$\delta(q_4, 1) = q_4$

Step 4: Construct the DFA



∴ The DFA can be defined as

$D = (Q, \Sigma, \delta, q_0, F)$

$Q = \{ q_0, q_1, q_2, q_3, q_4 \}$

$\Sigma = \{ 0, 1 \}$

$q_0 = \{ q_0 \}$

$F = \{ q_0 \}$

δ	0	1
$\rightarrow {}^*q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

Problem 23

Draw a DFA to accept decimal strings divisible by 3.

Solution

Step 1: Identify radix, Input alphabets and divisor.

$r = 10$ $d = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

$k = 3$

Step 2: Compute possible remainders

$i = 0, 1, 2$

Step 3: Compute transitions

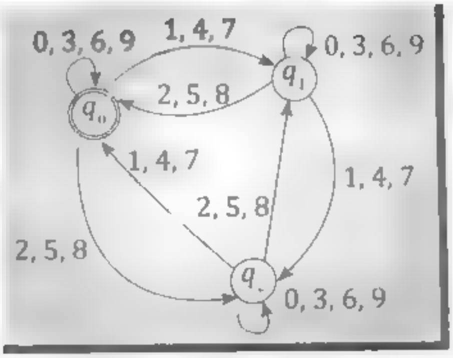
$\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$.

Here $j = (10 * i + d) \bmod 3$

Note For convenience let us group the digits from 0 to 9 based on the remainders {0, 3, 6, 9} with 0 as the remainder So δ from {0, 3, 6, 9} ⇒ δ from { 0 }
{1, 4, 7} with 1 as the remainder So δ from {1, 4, 7} ⇒ δ from { 1 }
{2, 5, 8} with 2 as the remainder So δ from {2, 5, 8} ⇒ δ from { 2 }.

Remainder	d	$(10 * i + d) \bmod 3 = j$	$\delta(q_i, d) = q_j$	
$i = 0$	0	$(10 * 0 + 0) \bmod 3 = 0$	$\delta(q_0, 0) = q_0$	$\Rightarrow \delta(q_0, \{0, 3, 6, 9\}) = q_0$
	1	$(10 * 0 + 1) \bmod 3 = 1$	$\delta(q_0, 1) = q_1$	$\Rightarrow \delta(q_0, \{1, 4, 7\}) = q_1$
	2	$(10 * 0 + 2) \bmod 3 = 2$	$\delta(q_0, 2) = q_2$	$\Rightarrow \delta(q_0, \{2, 5, 8\}) = q_2$
$i = 1$	0	$(10 * 1 + 0) \bmod 3 = 1$	$\delta(q_1, 0) = q_1$	$\Rightarrow \delta(q_1, \{0, 3, 6, 9\}) = q_1$
	1	$(10 * 1 + 1) \bmod 3 = 2$	$\delta(q_1, 1) = q_2$	$\Rightarrow \delta(q_1, \{1, 4, 7\}) = q_2$
	2	$(10 * 1 + 2) \bmod 3 = 0$	$\delta(q_1, 2) = q_0$	$\Rightarrow \delta(q_1, \{2, 5, 8\}) = q_0$
$i = 2$	0	$(10 * 2 + 0) \bmod 3 = 2$	$\delta(q_2, 0) = q_2$	$\Rightarrow \delta(q_2, \{0, 3, 6, 9\}) = q_2$
	1	$(10 * 2 + 1) \bmod 3 = 0$	$\delta(q_2, 1) = q_0$	$\Rightarrow \delta(q_2, \{1, 4, 7\}) = q_0$
	2	$(10 * 2 + 2) \bmod 3 = 1$	$\delta(q_2, 2) = q_1$	$\Rightarrow \delta(q_2, \{2, 5, 8\}) = q_1$

Step 5: Construct the DFA



The DFA can be defined as

Where

$D = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$q_0 = \{q_0\}$

$F = \{q_0\}$

δ	{0, 3, 6, 9}	{1, 4, 7}	{2, 5, 8}
$\rightarrow *q_0$	q_0	q_1	q_2
q_1	q_1	q_2	q
q_2	q_2	q_0	q_1

1.6.4.3 Modulo *k* Counter Problem

Problem 24

Draw a DFA to accept strings of even number of *a*'s

Solution

Step 1: $\Sigma = \{a\}$

Step 2: Identify the states

Since the string may have even number of *a*'s or odd number of *a*'s, the DFA can have two states.

(a) Even number of *a*'s denoted by $\textcircled{E}$

(b) Odd number of *a*'s denoted by $\textcircled{O}$

Step 3: Identify start state

Before reading any input, the number of *a*'s will be zero which represents Even *a*'s so $\textcircled{E}$ is the starting state.

$q_0 = \{E\}$

Step 4: Identify final state

Since the string should have even number of *a*'s, $\textcircled{E}$ is the final state.

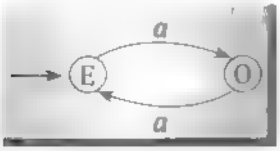
$F = \{E\}$

Step 5: Identify transitions

δ (E, a) = O reading a's from Even state leads to odd state

δ (O, a) = E reading a's from Odd state leads to even state

Step 6: Construct the DFA



∴ DFA is defined as

D = ($Q, \Sigma, \delta, q_0, F$)

$Q = \{ E, O \}$

$\Sigma = \{ a \}$

$q_0 = \{ E \}$

$F = \{ E \}$

δ	a
→* E	O
O	E

PROBLEM 25

Obtain a DFA to accept strings of a's and b's having even number of a's and even number of b's.

SOLUTION

Step 1: $\Sigma = \{ a, b \}$

Step 2: Identify the states

Let q_0 represents even number of a's and even number of b's.

q_1 - represents even number of a's and odd number of b's

q_2 - represents odd number of a's and even number of b's

q_3 - represents odd number of a's and odd number of b's

Step 3: $q_0 = \{ q_0 \}$

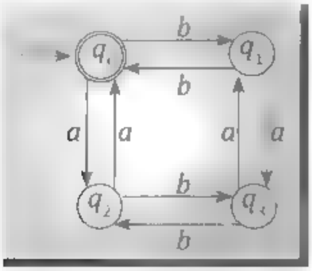
Step 4: $F = \{ q_0 \}$

Step 5: Identify transitions

Reading input from even state leads to odd state and vice versa, the transitions are

$\delta (q_0, a) = q_2$	$\delta (q_0, b) = q_1$
$\delta (q_1, a) = q_3$	$\delta (q_1, b) = q_0$
$\delta (q_3, a) = q_1$	$\delta (q_3, b) = q_2$

Step 4: Construct the DFA



∴ DFA is defined as

D = ($Q, \Sigma, \delta, q_0, F$)

$Q = \{ q_0, q_1, q_2, q_3 \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ q_0 \}$

$F = \{ q_0 \}$

δ	a	b
→*q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

PROBLEM 26

Obtain a DFA to accept strings of *a*'s and *b*'s having odd number of *a*'s and odd number of *b*'s.

SOLUTION

Step 1: $\Sigma = \{ a, b \}$

Step 2: Identify the states

Let q_0 - represents even number of a's and even number of b's.

q_1 - represents even number of a's and odd number of b's

q_2 - represents odd number of a's and even number of b's

q_3 - represents odd number of a's and odd number b's

Step 3: $q_0 = \{q_0\}$

Step 4: $F = \{q_3\}$

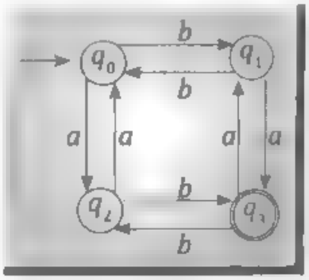
Step 5: Identify transitions

Reading input from even states leads to odd state and vice versa, the transitions are

$\delta(q_0, a) = q_2$ $\quad\delta(q_0, b) = q_1$ $\quad\delta(q_1, a) = q_3$

$\delta(q_1, b) = q_0$ $\quad\delta(q_3, a) = q_1$ $\quad\delta(q_3, b) = q_2$

Step 6: Construct the DFA



∴ DFA is defined as

$D = (Q, \Sigma, \delta, q_0, F)$

$Q = \{ q_0, q_1, q_2, q_3 \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ q_0 \}$

$F = \{ q_3 \}$

δ	a	b
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
$*q_3$	q_1	q_2

Note
- The DFA to accept even number of a's and odd number of b's can be obtained by making q1 as the only final state
- The DFA to accept odd number of a's and even number of b's can be obtained by making q2 as the only final state

PROBLEM 27

Obtain DFA to accept strings whose length is multiple of 3 on $\Sigma = \{ a, b \}$

(i.e.,) $L = \{ w \mid |w| \bmod 3 = 0 \}$

SOLUTION

Step 1: $\Sigma = \{ a, b \}$

Step 2: Identify the states

Since the length of the string should be divisible by 3, let the states be

q_0 - represents remainder 0.

q_1 - represents remainder 1.

q_2 - represents remainder 2.

Step 3: $q_0 - \{ q_0 \}$

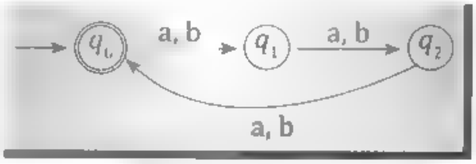
Step 4: $F - \{ q_0 \}$

Step 5: Identify transitions

$\delta(q_0, a) = q_1$ $\delta(q_0, b) = q_1$ $\delta(q_1, a) = q_2$

$\delta(q_1, b) = q_2$ $\delta(q_2, a) = q_0$ $\delta(q_2, b) = q_0$

Construct the DFA



$\therefore$ DFA is defined as

$\mathbf{D} = (Q, \Sigma, \delta, q_0, F)$

$Q = \{ q_0, q_1, q_2 \}$

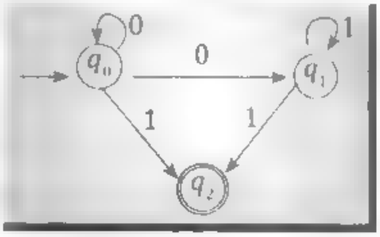
$\Sigma = \{ a, b \}$ $q_0 = \{ q_0 \}$

$F = \{ q_0 \}$

δ	a	b
$\rightarrow {}^*q_0$	q_1	q_1
q_1	q_2	q_2
q_2	q_0	q_0

1.7 – Non-Deterministic Finite Automata (NFA)

If an automata makes more than one transition for a single input symbol from a single state, then the next state of the automata becomes unpredictable Such type of automata is said to be NFA. If the finite automata is modified in such a way, that from a state on an input symbol, zero or more transitions are permitted, then the corresponding finite automata is called a NFA.



For example, consider the following finite automata

If the automation is in state q_0 and the input symbol is 0, what will be the next state? From the figure it is clear that the next state will be either q_0 or q_1. Thus some moves of the machine cannot be determined uniquely by the input symbol from the present state, such machines are called NFA.

Mathematical Representation of NFA

Definition: NFA is a five tuple, $(Q, \Sigma, \delta, q_0, F)$

Where Q is a non-empty finite set of states.

Σ is a non-empty finite set of input symbols.

$q_0 \subseteq Q$, is the start state

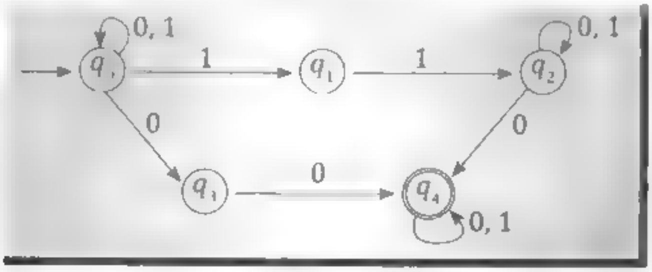
$F \subseteq Q$, is the set of accepting/final states

$\delta : Q \times \Sigma \longrightarrow 2^Q$, is the transition function.

2^Q is the power set of Q, which is the set of all subsets of Q.

Note In an NFA there can be zero, one or more transition on an input symbol.

Example :



This NFA is defined as

$N = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{q_0, q_1, q_2, q_3, q_4\}$

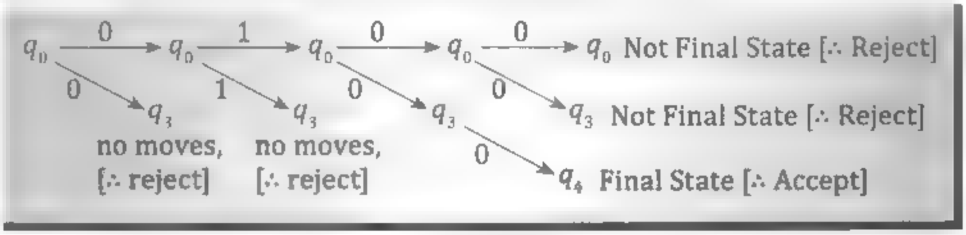
$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_4\}$

δ	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_2, q_4\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$
$*\ q_4$	$\{q_4\}$	$\{q_4\}$

The moves made by the above NFA for the string 0100 is as follows



q_0
0
q_0
1
q_0
0
q_0
0
q_0 Not Final State [∴ Reject]
0
q_3
1
q_3
0
q_3
0
q_3 Not Final State [∴ Reject]
no moves, [∴ reject]
no moves, [∴ reject]
0
q_4 Final State [∴ Accept]

The moves,

$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_3 \xrightarrow{0} q_4$ is accepted, since q_4 is the final state.

1.7.1 NFA Design

Language accepted by a NFA

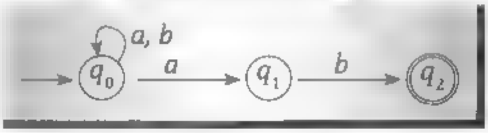
Let N = $(Q, \Sigma, \delta, q_0, F)$ be a NFA. A string w is accepted by the machine N, if it takes the initial state 'q_0' to final state. i.e., $\delta^*(q_0, w)$ is in F.

$L(N) = \{ w | w \in \Sigma^* \text{ and } \delta^*(q_0, w) \text{ is in } F$

PROBLEM 1

Draw NFA to accept strings of *a*'s and *b*'s ending with *ab*. Check whether the strings *abaab* and *abb* is accepted by the NFA.

SOLUTION



NFA is defined as

$N = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_2\}$

δ is shown by transition table

δ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

(i) For the string *abaab*, the moves are as follows.

$$\begin{tikzcd}
q_0 \arrow[r, "a"] \arrow[dr, "a"'] & q_0 \arrow[r, "b"] & q_0 \arrow[r, "a"] \arrow[dr, "a"] & q_0 \arrow[r, "a"] \arrow[dr, "a"] & q_0 \arrow[r, "b"] & q_0 \text{ Not Final State } [\therefore \text{Reject}] \\
 & q_0 \arrow[r, "b"] & q_0 \text{ no moves, } [\therefore \text{Reject}] & q_1 \; [\text{Reject}] & q_1 \arrow[dr, "b"] & \\
 & & & & & q_2 \text{ Final State } [\therefore \text{Accept}]
\end{tikzcd}$$

The moves, $q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2$ is accepted, since q_2 is the final state.

(ii) For the string *abb* the moves are as follows

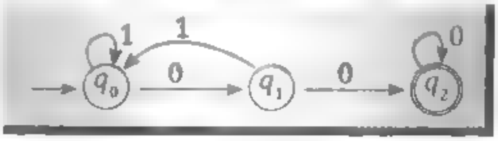
$$\begin{tikzcd}
q_0 \arrow[r, "a"] \arrow[dr, "a"'] & q_0 \arrow[r, "b"] & q_0 \arrow[r, "b"] & q_0 \text{ Not Final State } [\therefore \text{Reject}] \\
 & q_1 \arrow[r, "b"] & q_2 \; [\text{reject}] &
\end{tikzcd}$$

The string *abb* is rejected by the NFA, since q_0 is not a final state.

PROBLEM 2

Design an NFA that accepts a set of all strings ending in 00 and check whether the string 100 is accepted by the NFA.

SOLUTION



The NFA, $N = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_2\}$

δ	0	1
$\rightarrow q_0$	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_0\}$
$*q_2$	$\{q_2\}$	$\emptyset$

For the string 100 the moves are as follows.

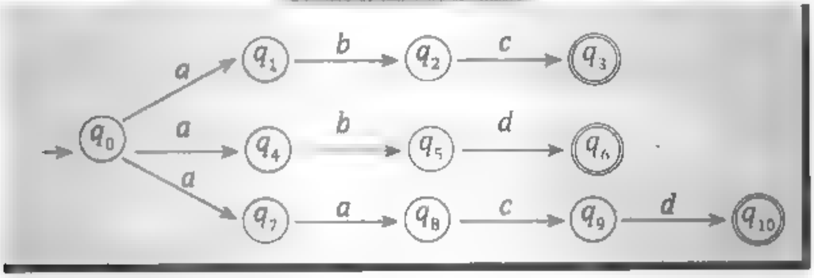
$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2$ [Final state ∴ accept]

The string 100 is accepted, since q_2 is the final state.

PROBLEM 3

Design a NFA to recognize the following set of strings *abc*, *abd* and *aacd*.

SOLUTION



The NFA,

$N = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}\}$

$\Sigma = \{a, b, c, d\}$

$q_0 = \{q_0\}$

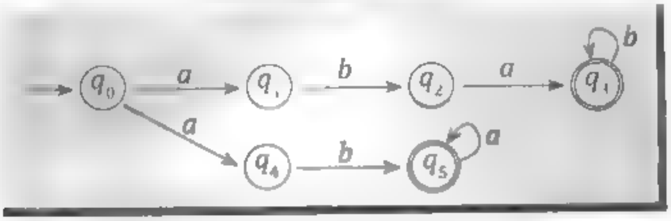
$F = \{q_3, q_6, q_{10}\}$

δ	a	b	c	d
$\rightarrow q_0$	$\{q_1, q_4, q_7\}$	$\emptyset$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\{q_2\}$	$\emptyset$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$*q_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_4	$\emptyset$	$\{q_5\}$	$\emptyset$	$\emptyset$
q_5	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_6\}$
$*q_6$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_7	$\{q_8\}$	$\emptyset$	$\emptyset$	$\emptyset$
q_8	$\emptyset$	$\emptyset$	$\{q_9\}$	$\emptyset$
q_9	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_{10}\}$
$*q_{10}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

PROBLEM 4

Obtain a NFA to accept the following language. $L = \{ w | w \in abab^n$ or aba^n where $n \geq 0 \}$

SOLUTION



$\therefore$ The NFA $N = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{ q_0, q_1, q_2, q_3, q_4, q_5 \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ q_0 \}$

$F = \{ q_3, q_5 \}$

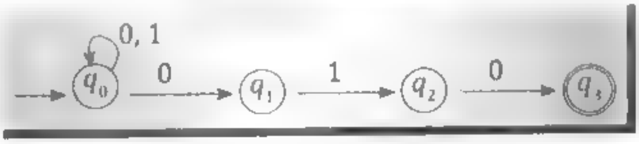
δ is shown using the transition table

δ	a	b
$\rightarrow q_0$	$\{ q_1, q_4 \}$	$\emptyset$
q_1	$\emptyset$	$\{ q_2 \}$
q_2	$\{ q_3 \}$	$\emptyset$
$*q_3$	$\emptyset$	$\{ q_3 \}$
q_4	$\emptyset$	$\{ q_5 \}$
$*q_5$	$\{ q_5 \}$	$\emptyset$

PROBLEM 5

Design a NFA to accept all the strings over the alphabet { 0, 1 } ending with 010.

SOLUTION



$\therefore$ The NFA, $N = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{ q_0, q_1, q_2, q_3 \}$

$\Sigma = \{ 0, 1 \}$

$q_0 = \{ q_0 \}$

$F = \{ q_3 \}$

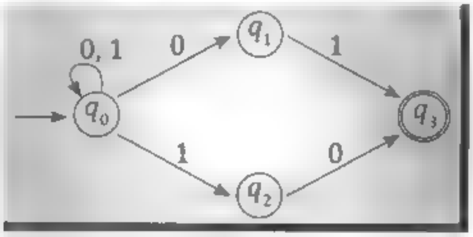
δ is shown using the transition table.

δ	0	1
$\rightarrow q_0$	$\{ q_0, q_1 \}$	$\{ q_0 \}$
q_1	$\emptyset$	$\{ q_2 \}$
q_2	$\{ q_3 \}$	$\emptyset$
$*q_3$	$\emptyset$	$\emptyset$

PROBLEM 6

Construct a NFA that accepts the set of all strings over {0, 1} that starts with 0 or 1 and ends with 01 or 10.

SOLUTION



∴ The NFA N = $(Q, \Sigma, \delta, q_0, F)$

Where $Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is shown using the transition table.

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\emptyset$	$\{q_3\}$
q_2	$\{q_3\}$	$\emptyset$
$*q_3$	$\emptyset$	$\emptyset$

1.7.2 Construction of DFA from NFA

Let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ be a NFA, which accepts the language L (N). Then there exists an equivalent DFA, $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ such that $L(N) = L(D)$.

Conversion of NFA to DFA can be done using subset construction or Lazy evaluation method. In subset construction method, if the NFA has n states then the transitions for all the 2^n states has to be found. Often all these states are not accessible from the start state of D. Inaccessible states can be thrown away. So effectively the number of states of D may be much smaller than 2^n. Hence the time complexity to convert NFA to DFA is $\Sigma \times 2^n$. The time complexity is exponential and the procedure takes long time. In order to avoid this we go for lazy evaluation method. In this method the transitions for all 2^n states are not found. Instead for each new state generated the corresponding transitions are found.

The steps to convert an NFA to DFA are as follows.

Step 1: The start state of NFA, is the start state of DFA. So add q_0 (which is the start stat of N) to Q_D and find the transitions from this state.

Step 2: For each state $\{q_1, q_2, q_3, \_\_\_\_\_\_ q_r\}$ in Q_D, the transitions for each input symbol in Σ can be obtained as follows.

(a) $\delta_D(\{q_1, q_2, q_3, \_\_\_ q_i\}, a) = \delta_N(q_1, a) \cup \delta_N(q_2, a) \cup \cdots \delta_N(q_i, a) = \{q_1, q_2, \_\_\_ q_k\}$

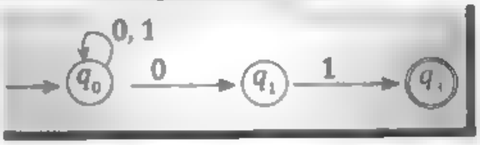
(b) Add the state $\{q_1, q_2, \ldots\ldots q_k\}$ to Q_D, if it is not already in Q_D, repeat step 2.

(c) If no new states are generated then stop.

Step 3: The state $\{q_1, q_2, \ldots\ldots q_n\}$ which belongs to Q_D is the final state, if atleast one of the state in $q_1, q_2, \ldots\ldots q_n$ is a final state of NFA.

Problem 1

Convert the following NFA into an equivalent DFA



Solution

The given NFA, N = $(Q_N, \Sigma, \delta_N, q_0, F_N)$

Where $Q_N = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F_N = \{q_2\}$

δ_N	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

The equivalent DFA, $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{0, 1\}$

Step 1: q_0 is the start state of DFA, since q_0 is the start state of NFA. $\therefore Q_D = \{q_0\}$

Step 2: Obtain the corresponding transitions for each state in Q_D on all input symbols. If there are any new states add it to Q_D and find the transitions.

δ_D from q_0 on Σ

$\delta_D(q_0, 0) = \delta_N(q_0, 0)$

$= \{q_0, q_1\}$

$\delta_D(q_0, 1) = \delta_N(q_0, 1) = \{q_0\}$

Now we have found out a new state $\therefore$ find δ_D from this new state. $\{q_0, q_1\}$.

δ_D from $\{q_0, q_1\}$ on Σ

$\delta_D(\{q_0, q_1\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

$= \{q_0, q_1\} \cup \emptyset$

$= \{q_0, q_1\}$

$\delta_D(\{q_0, q_1\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$

$= \{q_0\} \cup \{q_2\}$

$= \{q_0, q_2\}$

Again we have found a new state $\{q_0, q_2\}$

δ_D from $\{q_0, q_2\}$ on Σ

$$\begin{aligned}\delta_D(\{q_0, q_2\}, 0) &= \delta_N(q_0, 0) \cup \delta_N(q_2, 0)\\ &= \{q_0, q_1\} \cup \emptyset\\ &= \{q_0, q_1\}\\ \delta_D(\{q_0, q_2\}, 1) &= \delta_N(q_0, 1) \cup \delta_N(q_2, 1)\\ &= \{q_0\} \cup \emptyset\\ &= \{q_0\}\end{aligned}$$

No more new states therefore we stop.

Step 3: The final state, F_D is the states in Q_D which has q_2 [which is the final state of F_N]

$F_D = \{q_0, q_2\}$

$\therefore$ The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

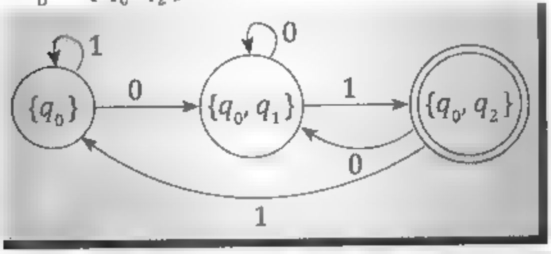
Where $Q_D = \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}\}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F_D = \{q_0, q_2\}$

δ_D	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$* \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

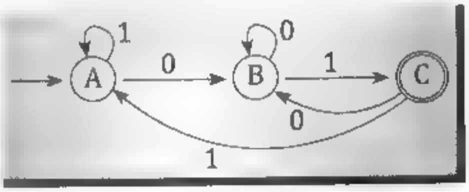


Note The states can be renamed as follows Let A = $\{q_0\}$, B = $\{q_0, q_1\}$, C = $\{q_0, q_2\}$

The transition table can be rewritten as follows.

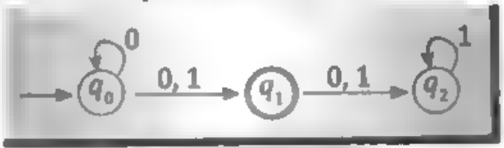
δ_D	0	1
$\rightarrow$ A	B	A
B	B	C
*C	B	A

The equivalent DFA



PROBLEM 2

Convert the following NFA to its equivalent DFA



SOLUTION

The given NFA, N $= (Q_N, \Sigma, \delta_N, q_0, F_N)$

Where $Q_N = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F_N = \{q_1\}$

δ_N	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$*q_1$	$\{q_2\}$	$\{q_2\}$
q_2	$\emptyset$	$\{q_2\}$

The equivalent DFA, D $= (Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{0, 1\}$

Step 1: $q_0 = \{q_0\}$, q_0 is the start state of DFA, D.

Step 2: δ_D from q_0 on Σ

$\delta_D(q_0, 0) = \delta_N(q_0, 0)$
$= \{q_0, q_1\}$

$\delta_D(q_0, 1) = \delta_N(q_0, 1)$
$= \{q_1\}$

δ_D from $\{q_0, q_1\}$ on Σ

$\delta_D(\{q_0, q_1\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$
$= \{q_0, q_1\} \cup \{q_2\}$
$= \{q_0, q_1, q_2\}$

$\delta_D(\{q_0, q_1\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$
$= \{q_1\} \cup \{q_2\}$
$= \{q_1, q_2\}$

δ_D from $\{q_1\}$ on Σ

$\delta_D(\{q_1\}, 0) = \delta_N(q_1, 0)$
$= \{q_2\}$

$\delta_D(\{q_1\}, 1) = \delta_N(q_1, 1)$
$= \{q_2\}$

δ_D from $\{q_0, q_1, q_2\}$ on Σ

$\delta_D(\{q_0, q_1, q_2\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_0, q_1\} \cup \{q_2\} \cup \emptyset$

$= \{q_0, q_1, q_2\}$

$\delta_D(\{q_0, q_1, q_2\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1) \cup \delta_N(q_2, 1)$

$= \{q_1\} \cup \{q_2\} \cup \{q_2\}$

$= \{q_1, q_2\}$

δ_D from $\{q_1, q_2\}$ on Σ

$\delta_D(\{q_1, q_2\}, 0) = \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_2\} \cup \emptyset$

$= \{q_2\}$

$\delta_D(\{q_1, q_2\}, 1) = \delta_N(q_1, 1) \cup \delta_N(q_2, 1)$

$= \{q_2\} \cup \{q_2\}$

$= \{q_2\}$

δ_D from $\{q_2\}$ on Σ

$\delta_D(\{q_2\}, 0) = \delta_N(q_2, 0)$

$= \emptyset$

$\delta_D(\{q_2\}, 1) = \delta_N(q_2, 1)$

$= \{q_2\}$

Step 3: The final state $F_D = \{\{q_0, q_1\}, \{q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}\}$

$\therefore$ The equivalent **DFA $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$**

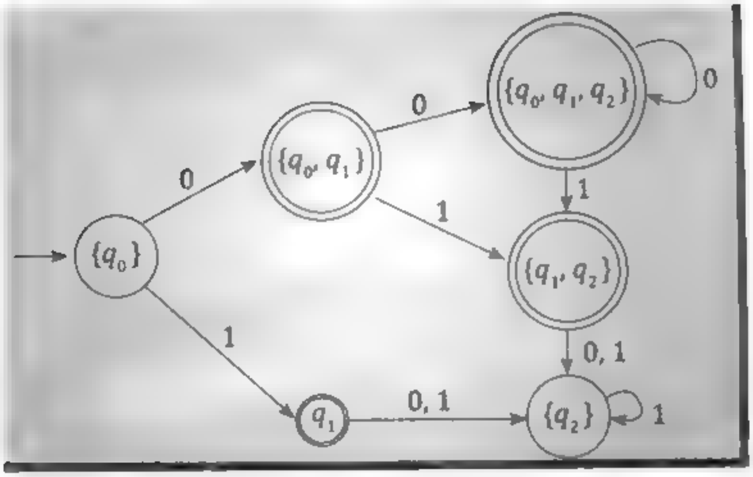
Where, $Q_D = \{\{q_0\}, \{q_0, q_1\}, \{q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\}\}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F_D = \{\{q_0, q_1\}, \{q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}\}$

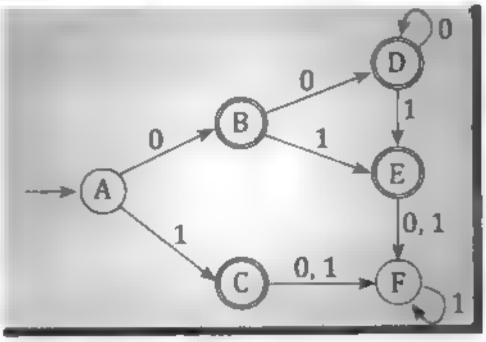
δ_D	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_1\}$
$* \{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
$* \{q_1\}$	$\{q_2\}$	$\{q_2\}$
$* \{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
$* \{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_2\}$	$\emptyset$	$\{q_2\}$



After renaming the states the transition table is as follows

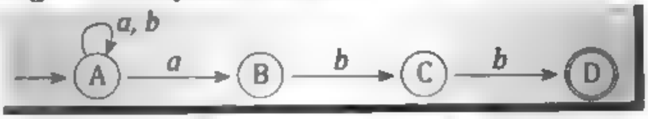
δ_D	0	1
→ A	{ B }	{ C }
*B	{ D }	{ E }
*C	{ F }	{ F }
*D	{ D }	{ E }
*E	{ F }	{ F }
F	∅	{ F }

The equivalent DFA



PROBLEM 3

Convert the following NFA to equivalent DFA.



SOLUTION

The given NFA N = $(Q_N, \Sigma, \delta_N, q_0, F_N)$

Where $Q_N = \{A, B, C, D\}$

$\Sigma = \{a, b\}$

$q_0 = A$

$F_N = \{D\}$

δ_N	a	b
→ A	{ A, B }	{ A }
B	∅	{ C }
C	∅	{ D }
*D	∅	∅

The equivalent DFA D = $(Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{ 0, 1 \}$

Step 1: $q_0 = \{ A \}$

Step 2: **δ_D from { A } on Σ**

$\delta_D (A, a) = \delta_N (A, a) = \{ A, B \}$

$\delta_D (A, b) = \delta_N (A, b) = \{ A \}$

δ_D from { A, B } on Σ

$$\begin{aligned}\delta_D (\{ A, B\}, a) &= \delta_N (A, a) \cup \delta_N (B, a)\\ &= \{ A, B \} \cup \emptyset\\ &= \{ A, B \}\end{aligned}$$

$$\begin{aligned}\delta_D (\{ A, B\}, b) &= \delta_N (A, b) \cup \delta_N (B, b)\\ &= \{ A \} \cup \{ C \}\\ &= \{ A, C \}\end{aligned}$$

δ_D from { A, C } on Σ

$$\begin{aligned}\delta_D (\{ A, C\}, a) &= \delta_N (A, a) \cup \delta_N (C, a)\\ &= \{ A, B \} \cup \emptyset\\ &= \{ A, B \}\end{aligned}$$

$$\begin{aligned}\delta_D (\{ A, C\}, b) &= \delta_N (A, b) \cup \delta_N (C, b)\\ &= \{ A \} \cup \{ D \}\\ &= \{ A, D \}\end{aligned}$$

δ_D from { A, D } on Σ

$$\begin{aligned}\delta_D (\{ A, D\}, a) &= \delta_N (A, a) \cup \delta_N (D, a)\\ &= \{ A, B \} \cup \emptyset\\ &= \{ A, B \}\end{aligned}$$

$$\begin{aligned}\delta_D (\{ A, D\}, b) &= \delta_N (A, b) \cup \delta_N (D, b)\\ &= \{ A \} \cup \emptyset\\ &= \{ A \}\end{aligned}$$

Step 3: The final state

$F_D = \{ \{A, D\} \}$

∴ The equivalent DFA.

D = $(Q_D, \Sigma, \delta_D, q_0, F_D)$

Where

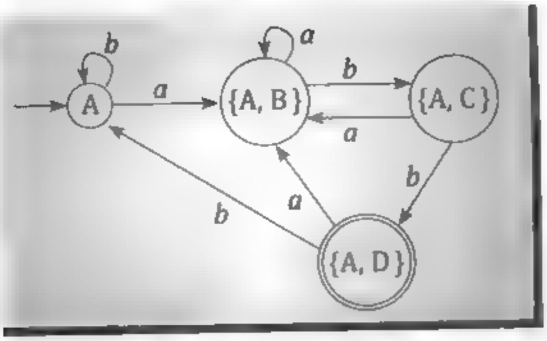
$Q_D = \{ \{ A \}, \{ A, B \}, \{ A, C \}, \{ A, D \} \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ A \}$

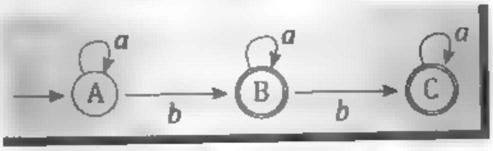
$F_D = \{ A, D \}$

δ_D	a	b
→ { A }	{ A, B }	{ A }
{ A, B }	{ A, B }	{ A, C }
{ A, C }	{ A, B }	{ A, D }
*{ A, D }	{ A, B }	{ A }



Problem 4

Convert the given NFA to an equivalent DFA.



Solution

The given NFA, N = $(Q_N, \Sigma, \delta_N, q_0, F_N)$

Where $Q_N = \{A, B, C\}$

$\Sigma = \{a, b\}$

$q_0 = \{A\}$

$F_N = \{B, C\}$

δ_N	a	b
$\rightarrow$ A	{A}	{B}
* B	{B}	{C}
* C	{C}	Ø

The equivalent DFA D = $(Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{a, b\}$

Step 1: $q_0 = \{A\}$

Step 2: δ_D from A on Σ

$\delta_D(A, a) = \delta_N(A, a) = \{A\}$

$\delta_D(A, b) = \delta_N(A, b) = \{B\}$

δ_D from B on Σ

$\delta_D(B, a) = \delta_N(B, a) = \{B\}$

$\delta_D(B, b) = \delta_N(B, b) = \{C\}$

δ_D from {C} on Σ

$\delta_D(C, a) = \delta_N(C, a)$

$= \{C\}$

$\delta_D(C, b) = \delta_N(C, b)$

$= \varnothing$

Step 3: The final state

$F_D = \{B, C\}$

∴ The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

Where

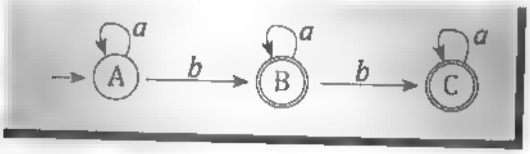
$Q_D = \{A, B, C\}$

$\Sigma = \{a, b\}$

$q_0 = A$

$F_D = \{B, C\}$

δ_D	a	b
→ A	{ A }	{ B }
* B	{ B }	{ C }
* C	{ C }	∅



Note DFA and NFA are the same.

PROBLEM 5

Convert the following NFA to DFA.

δ_D	0	1
→ P	$\{P, Q\}$	$\{P\}$
Q	$\{R\}$	$\{R\}$
R	$\{S\}$	∅
$*S$	$\{S\}$	$\{S\}$

SOLUTION

The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{0, 1\}$

Step 1: $q_0 = P$

Step 2: δ_D from $\{P\}$ on Σ

$\delta_D(P, 0) = \delta_N(P, 0) = \{P, Q\}$

$\delta_D(P, 1) = \delta_N(P, 1) = \{P\}$

δ_D from $\{P, Q\}$ on Σ

$$\begin{aligned}\delta_D(\{P, Q\}, 0) &= \delta_N(P, 0) \cup \delta_N(Q, 0)\\ &= \{P, Q\} \cup \{R\}\\ &= \{P, Q, R\}\end{aligned}$$

$$\begin{aligned}\delta_D(\{P, Q\}, 1) &= \delta_N(P, 1) \cup \delta_N(Q, 1)\\ &= \{P\} \cup \{R\}\\ &= \{P, R\}\end{aligned}$$

δ_D from $\{P, Q, R\}$ on Σ

$$\begin{aligned}\delta_D(\{P, Q, R\}, 0) &= \delta_N(P, 0) \cup \delta_N(Q, 0) \cup \delta_N(R, 0)\\ &= \{P, Q\} \cup \{R\} \cup \{S\}\\ &= \{P, Q, R, S\}\end{aligned}$$

$$\begin{aligned}\delta_D(\{P, Q, R\}, 1) &= \delta_N(P, 1) \cup \delta_N(Q, 1) \cup \delta_N(R, 1)\\ &= \{P\} \cup \{R\} \cup \emptyset\\ &= \{P, R\}\end{aligned}$$

δ_D from $\{P, R\}$ on $\sum$

$$\begin{aligned}\delta_D(\{P, R\}, 0) &= \delta_N(P, 0) \cup \delta_N(R, 0)\\ &= \{P, Q\} \cup \{S\}\\ &= \{P, Q, S\}\end{aligned}$$

$$\begin{aligned}\delta_D(\{P, R\}, 1) &= \delta_N(P, 1) \cup \delta_N(R, 1)\\ &= \{P\} \cup \emptyset\\ &= \{P\}\end{aligned}$$

δ_D from $\{P, Q, R, S\}$ on $\sum$

$$\begin{aligned}\delta_D(\{P, Q, R, S\}, 0) &= \delta_N(P, 0) \cup \delta_N(Q, 0) \cup \delta_N(R, 0) \cup \delta_N(S, 0)\\ &= \{P, Q\} \cup \{R\} \cup \{S\} \cup \{S\}\\ &= \{P, Q, R, S\}\end{aligned}$$

$$\begin{aligned}\delta_D(\{P, Q, R, S\}, 1) &= \delta_N(P, 1) \cup \delta_N(Q, 1) \cup \delta_N(R, 1) \cup \delta_N(S, 1)\\ &= \{P\} \cup \{R\} \cup \emptyset \cup \{S\}\\ &= \{P, R, S\}\end{aligned}$$

δ_D from $\{P, Q, S\}$ on $\sum$

$$\begin{aligned}\delta_D(\{P, Q, S\}, 0) &= \delta_N(P, 0) \cup \delta_N(Q, 0) \cup \delta_N(S, 0)\\ &= \{P, Q\} \cup \{R\} \cup \{S\}\\ &= \{P, Q, R, S\}\end{aligned}$$

$$\begin{aligned}\delta_D(\{P, Q, S\}, 1) &= \delta_N(P, 1) \cup \delta_N(Q, 1) \cup \delta_N(S, 1)\\ &= \{P\} \cup \{R\} \cup \{S\}\\ &= \{P, R, S\}\end{aligned}$$

δ_D from $\{P, R, S\}$ on Σ

$\delta_D(\{P, R, S\}, 0) = \delta_N(P, 0) \cup \delta_N(R, 0) \cup \delta_N(S, 0)$

$= \{P, Q\} \cup \{S\} \cup \{S\}$

$= \{P, Q, S\}$

$\delta_D(\{P, R, S\}, 1) = \delta_N(P, 1) \cup \delta_N(R, 1) \cup \delta_N(S, 1)$

$= \{P\} \cup \emptyset \cup \{S\}$

$= \{P, S\}$

δ_D from $\{P, S\}$ on Σ

$\delta_D(\{P, S\}, 0) = \delta_N(P, 0) \cup \delta_N(S, 0)$

$= \{P, Q\} \cup \{S\}$

$= \{P, Q, S\}$

$\delta_D(\{P, S\}, 1) = \delta_N(P, 1) \cup \delta_N(S, 1)$

$= \{P\} \cup \{S\}$

$= \{P, S\}$

Step 3: The final state

$F_D = \{\{P, Q, R, S\}, \{P, Q, S\}, \{P, R, S\}, \{P, S\}\}$

$\therefore$ The equivalent DFA

$\mathbf{D} = (Q_D, \Sigma, \delta_D, q_0, F_D)$

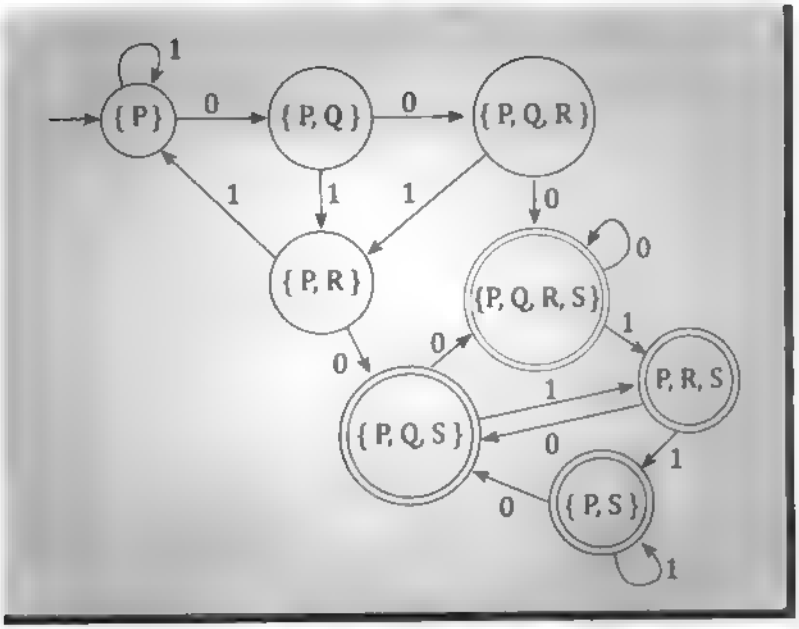
Where $Q_D = \{\{P\}, \{P, Q\}, \{P, Q, R\}, \{P, R\}, \{P, Q, R, S\}, \{P, Q, S\}, \{P, R, S\}, \{P, S\}$

$\Sigma = \{0, 1\}$

$q_0 = \{P\}$

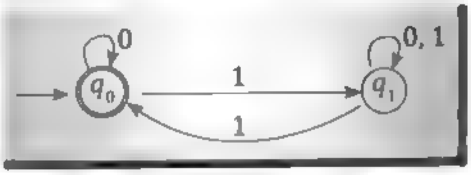
$F_D = \{\{P, Q, R, S\}, \{P, Q, S\}, \{P, R, S\}, \{P, S\}\}$

δ_D	0	1
$\rightarrow\{P\}$	$\{P, Q\}$	$\{P\}$
$\{P, Q\}$	$\{P, Q, R\}$	$\{P, R\}$
$\{P, Q, R\}$	$\{P, Q, R, S\}$	$\{P, R\}$
$\{P, R\}$	$\{P, Q, S\}$	$\{P\}$
$*\{P, Q, R, S\}$	$\{P, Q, R, S\}$	$\{P, R, S\}$
$*\{P, Q, S\}$	$\{P, Q, R, S\}$	$\{P, R, S\}$
$*\{P, R, S\}$	$\{P, Q, S\}$	$\{P, S\}$
$*\{P, S\}$	$\{P, Q, S\}$	$\{P, S\}$



— Problem 6 —

Convert the following NFA to DFA



Solution

The given NFA, N = $(Q_N, \Sigma, \delta_N, q_0, F_N)$

Where $Q_N = \{ q_0, q_1 \}$

$\Sigma = \{ 0, 1 \}$

$q_0 = \{ q_0 \}$

$F_N = \{ q_0 \}$

δ_N	0	1
$\rightarrow *q_0$	$\{ q_0 \}$	$\{ q_1 \}$
q_1	$\{ q_1 \}$	$\{ q_0, q_1 \}$

The equivalent DFA D = $(Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{ 0, 1 \}$

Step 1: $q_0 = \{ q_0 \}$

Step 2: **δ_D from $\{ q_0 \}$ on Σ**

$$\begin{aligned}\delta_D(q_0, 0) &= \delta_N(q_0, 0)\\ &= \{ q_0 \}\\ \delta_D(q_0, 1) &= \delta_N(q_0, 1)\\ &= \{ q_1 \}\end{aligned}$$

δ_D from $\{ q_1 \}$ on Σ

$$\begin{aligned}\delta_D(q_1, 0) &= \delta_N(q_1, 0)\\ &= \{ q_1 \}\\ \delta_D(q_1, 1) &= \delta_N(q_1, 1)\\ &= \{ q_0, q_1 \}\end{aligned}$$

δ_D from $\{ q_0, q_1 \}$ on Σ

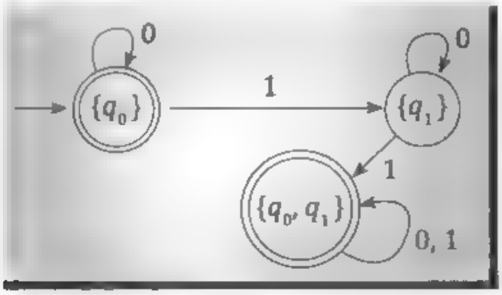
$$\begin{aligned}\delta_D(\{q_0, q_1\}, 0) &= \delta_N(q_0, 0) \cup \delta_N(q_1, 0)\\ &= \{ q_0 \} \cup \{ q_1 \}\\ &= \{ q_0, q_1 \}\\ \delta_D(\{q_0, q_1\}, 1) &= \delta_N(q_0, 1) \cup \delta_N(q_1, 1)\\ &= \{ q_1 \} \cup \{ q_0, q_1 \}\\ &= \{ q_0, q_1 \}\end{aligned}$$

Step 3: The final state $F_D = \{ \{ q_0 \}, \{ q_0, q_1 \} \}$

∴ The equivalent DFA,

$$\begin{aligned}D &= (Q_D, \Sigma, \delta_D, q_0, F_D)\\ \Sigma &= \{ 0, 1 \}\\ q_0 &= \{ q_0 \}\\ F_D &= \{ \{ q_0 \}, \{ q_0, q_1 \} \}\end{aligned}$$

δ_D	0	1
$\rightarrow * \{ q_0 \}$	$\{ q_0 \}$	$\{ q_1 \}$
$\{ q_1 \}$	$\{ q_1 \}$	$\{ q_0, q_1 \}$
$*\{ q_0, q_1 \}$	$\{ q_0, q_1 \}$	$\{ q_0, q_1 \}$



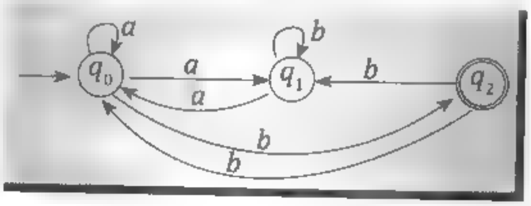
PROBLEM 7

Find a DFA equivalent to the following, N = ({ q_0, q_1, q_2 }, { a, b }, δ, q_0, { q_2 })
Where δ is defined as follows:

δ_D	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_1\}$
$*q_2$	-	$\{q_0, q_1\}$

SOLUTION

The transition diagram for the given NFA is



The equivalent DFA

$$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$$

Where $\Sigma = \{ a, b \}$

Step 1: $q_0 = \{ q_0 \}$

Step 2: δ_D from { q_0 } on Σ

$\delta_D (q_0, a) = \delta_N (q_0, a)$

$= \{ q_0, q_1 \}$

$\delta_D (q_0, b) = \delta_N (q_0, b)$

$= \{ q_2 \}$

δ_D from { q_0, q_1 } on Σ

$\delta_D (\{q_0, q_1\}, a) = \delta_N (q_0, a) \cup \delta_N (q_1, a)$

$= \{ q_0, q_1 \} \cup \{ q_0 \}$

$= \{ q_0, q_1 \}$

$\delta_D (\{q_0, q_1\}, b) = \delta_N (q_0, b) \cup \delta_N (q_1, b)$

$= \{ q_2 \} \cup \{ q_1 \}$

$= \{ q_1, q_2 \}$

δ_D from { q_2} on Σ

$\delta_D(q_2, a) = \delta_N(q_2, a)$

$= \emptyset$

$\delta_D(q_2, b) = \delta_N(q_2, b)$

$= \{q_0, q_1\}$

δ_D from { q_1, q_2} on Σ

$\delta_D(\{q_1, q_2\}, a) = \delta_N(q_1, a) \cup \delta_N(q_2, a)$

$= \{q_0\} \cup \emptyset$

$= \{q_0\}$

$\delta_D(\{q_1, q_2\}, b) = \delta_N(q_1, b) \cup \delta_N(q_2, b)$

$= \{q_1\} \cup \{q_0, q_1\}$

$= \{q_0, q_1\}$

Step 3: The final state $F_D = \{\{q_2\}, \{q_1, q_2\}\}$

The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

Where

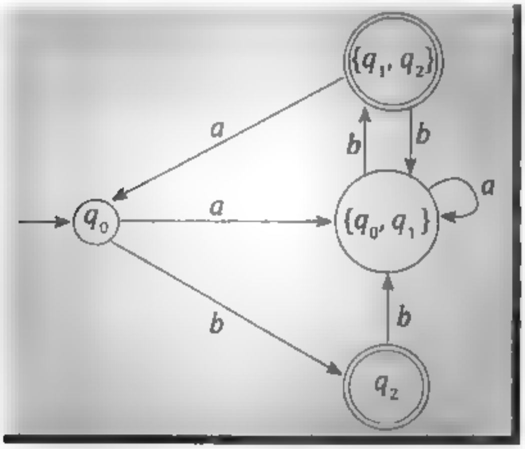
$Q_D = \{\{q_0\}, \{q_0, q_1\}, \{q_2\} \{q_1, q_2\}\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F_D = \{\{q_2\}, \{q_1, q_2\}\}$

δ_D	a	b
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_2\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$^*\{q_2\}$	$\emptyset$	$\{q_0, q_1\}$
$^*\{q_1, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$



— Problem 8 —

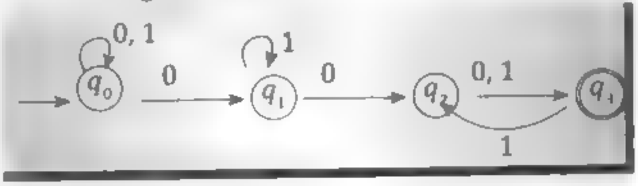
Construct a DFA equivalent to N = ($\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\}$)

Where δ is given by

δ_D	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_1\}$
q_2	$\{q_3\}$	$\{q_3\}$
$* q_3$	$\emptyset$	$\{q_2\}$

Solution

The transition diagram for the given NFA is



The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{0, 1\}$

Step 1: $q_0 = \{q_0\}$

Step 2: δ_D from $\{q_0\}$ on Σ

$\delta_D(q_0, 0) = \delta_N(q_0, 0)$

$= \{q_0, q_1\}$

$\delta_D(q_0, 1) = \delta_N(q_0, 1)$

$= \{q_0\}$

δ_D from $\{q_0, q_1\}$ on Σ

$\delta_D(\{q_0, q_1\}, 0) = \delta_N(q_1, 0) \cup \delta_N(q_1, 0)$

$= \{q_0, q_1\} \cup \{q_2\}$

$= \{q_0, q_1, q_2\}$

$\delta_D(\{q_0, q_1\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$

$= \{q_0\} \cup \{q_1\}$

$= \{q_0, q_1\}$

δ_D from $\{ q_0, q_1, q_2 \}$ on Σ

$\delta_D (\{q_0, q_1, q_2\}, 0) = \delta_N (q_0, 0) \cup \delta_N (q_1, 0) \cup \delta_N (q_2, 0)$
$= \{ q_0, q_1 \} \cup \{ q_2 \} \cup \{ q_3 \}$
$= \{ q_0, q_1, q_2, q_3 \}$

$\delta_D (\{q_0, q_1, q_2\}, 1) = \delta_N (q_0, 1) \cup \delta_N (q_1, 1) \cup \delta_N (q_2, 1)$
$= \{ q_0 \} \cup \{ q_1 \} \cup \{ q_3 \}$
$= \{ q_0, q_1, q_3 \}$

δ_D from $\{ q_0, q_1, q_2, q_3 \}$ on $\sum$

$\delta_D (\{q_0, q_1, q_2, q_3\}, 0) = \delta_N (q_0, 0) \cup \delta_N (q_1, 0) \cup \delta_N (q_2, 0) \cup \delta_N (q_3, 0)$
$= \{ q_0, q_1 \} \cup \{ q_2 \} \cup \{ q_3 \} \cup \emptyset$
$= \{ q_0, q_1, q_2, q_1 \}$

$\delta_D (\{q_0, q_1, q_2, q_3\}, 1) = \delta_N (q_0, 1) \cup \delta_N (q_1, 1) \cup \delta_N (q_2, 1) \cup \delta_N (q_3, 1)$
$= \{ q_0 \} \cup \{ q_1 \} \cup \{ q_3 \} \cup \{ q_2 \}$
$= \{ q_0, q_1, q_2, q_3 \}$

δ_D from $\{ q_0, q_1, q_3 \}$ on $\sum$

$\delta_D (\{q_0, q_1, q_3\}, 0) = \delta_N (q_0, 0) \cup \delta_N (q_1, 0) \cup \delta_N (q_3, 0)$
$= \{ q_0, q_1 \} \cup \{ q_2 \} \cup \emptyset$
$= \{ q_0, q_1, q_2 \}$

$\delta_D (\{q_0, q_1, q_3\}, 1) = \delta_N (q_0, 1) \cup \delta_N (q_1, 1) \cup \delta_N (q_3, 1)$
$= \{ q_0 \} \cup \{ q_1 \} \cup \{ q_2 \}$
$= \{ q_0, q_1, q_2 \}$

Step 3: The final state $F_D = \{ \{q_0, q_1, q_2, q_3\}, \{q_0, q_1, q_3\} \}$

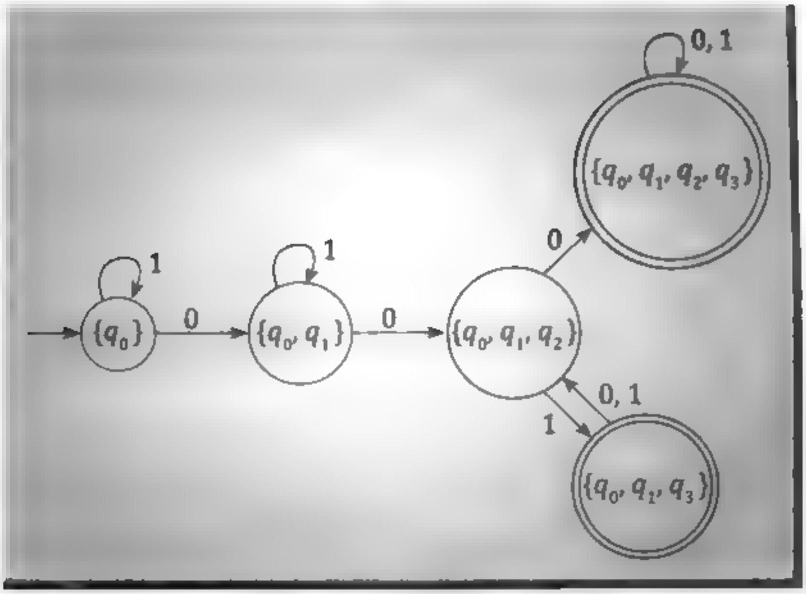
$\therefore$ The equivalent DFA

$D = (Q_D, \sum, \delta_D, q_0, F_D)$

Where

$Q_D = \{ \{ q_0 \}, \{ q_0, q_1 \}, \{ q_0, q_1, q_2 \}, \{ q_0, q_1, q_2, q_3 \}, \{ q_0, q_1, q_3 \} \}$
$\sum = \{ 0, 1 \}$
$q_0 = \{ q_0 \}$
$F_D = \{ \{q_0, q_1, q_2, q_3\}, \{q_0, q_1, q_3\} \}$

δ_D	0	1
$\rightarrow \{ q_0 \}$	$\{ q_0, q_1 \}$	$\{ q_0 \}$
$\{ q_0, q_1 \}$	$\{ q_0, q_1, q_2 \}$	$\{ q_0, q_1 \}$
$\{ q_0, q_1, q_2 \}$	$\{ q_0, q_1, q_2, q_3 \}$	$\{ q_0, q_1, q_3 \}$
* $\{ q_0, q_1, q_2, q_3 \}$	$\{ q_0, q_1, q_2, q_3 \}$	$\{ q_0, q_1, q_2, q_3 \}$
* $\{ q_0, q_1, q_3 \}$	$\{ q_0, q_1, q_2 \}$	$\{ q_0, q_1, q_2 \}$



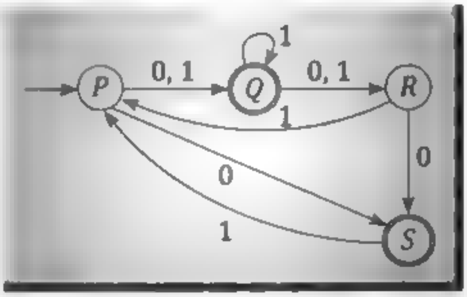
PROBLEM 9

Construct a DFA equivalent to $N = (\{P, Q, R, S\}, \{0, 1\}, \delta, P, \{Q, S\})$
Where δ is given by

	0	1
$\rightarrow P$	$\{Q, S\}$	$\{Q\}$
$*Q$	$\{R\}$	$\{Q, R\}$
R	$\{S\}$	$\{P\}$
$*S$	$\emptyset$	$\{P\}$

[illegible]

The transition diagram for the NFA, N is



The equivalent DFA

$$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$$
$$\Sigma = \{0, 1\}$$

Step 1: $q_0 = \{P\}$

Step 2: **δ_D from $\{P\}$ on Σ**

$\delta_D(P, 0) = \delta_N(P, 0)$
$= \{Q, S\}$

$\delta_D(P, 1) = \delta_D(P, 1)$
$= \{Q\}$

δ_D from $\{Q, S\}$ on Σ

$\delta_D(\{Q, S\}, 0) = \delta_N(Q, 0) \cup \delta_N(S, 0)$
$= \{R\} \cup \emptyset$
$= \{R\}$

$\delta_D(\{Q, S\}, 1) = \delta_N(Q, 1) \cup \delta_N(S, 1)$
$= \{Q, R\} \cup \{P\}$
$= \{P, Q, R\}$

δ_D from $\{Q\}$ on Σ

$\delta_D(Q, 0) = \delta_N(Q, 0)$
$= \{R\}$

$\delta_D(Q, 1) = \delta_N(Q, 1)$
$= \{Q, R\}$

δ_D from $\{R\}$ on Σ

$\delta_D(R, 0) = \delta_N(R, 0)$
$= \{S\}$

$\delta_D(R, 1) = \delta_N(R, 1) = \{P\}$

δ_D from $\{P, Q, R\}$ on Σ

$\delta_D(\{P, Q, R\}, 0) = \delta_N(P, 0) \cup \delta_N(Q, 0) \cup \delta_N(R, 0)$
$= \{Q, S\} \cup \{R\} \cup \{S\}$
$= \{Q, R, S\}$

$\delta_D(\{P, Q, R\}, 1) = \delta_N(P, 1) \cup \delta_N(Q, 1) \cup \delta_N(R, 1)$
$= \{Q\} \cup \{Q, R\} \cup \{P\}$
$= \{P, Q, R\}$

δ_D from $\{Q, R\}$ on Σ

$\delta_D(\{Q, R\}, 0) = \delta_N(Q, 0) \cup \delta_N(R, 0)$
$= \{R\} \cup \{S\}$
$= \{R, S\}$

$\delta_D(\{Q, R\}, 1) = \delta_N(Q, 1) \cup \delta_N(R, 1)$
$= \{Q, R\} \cup \{P\}$
$= \{P, Q, R\}$

δ_D from $\{S\}$ on Σ

$\delta_D(\{S\}, 0) = \delta_N(S, 0)$
$= \emptyset$

$\delta_D(\{S\}, 1) = \delta_N(S, 1)$
$= \{P\}$

δ_D from $\{Q, R, S\}$ on Σ

$\delta_D(\{Q, R, S\}, 0) = \delta_N(Q, 0) \cup \delta_N(R, 0) \cup \delta_N(S, 0)$
$= \{R\} \cup \{S\} \cup \emptyset$
$= \{R, S\}$

$\delta_D(\{Q, R, S\}, 1) = \delta_N(Q, 1) \cup \delta_N(R, 1) \cup \delta_N(S, 1)$
$= \{Q, R\} \cup \{P\} \cup \{P\}$
$= \{P, Q, R\}$

δ_D from $\{R, S\}$ on $\sum$

$\delta_D(\{R, S\}, 0) = \delta_N(R, 0) \cup \delta_N(S, 0)$
$= \{S\} \cup \emptyset$
$= \{S\}$

$\delta_D(\{R, S\}, 1) = \delta_N(R, 1) \cup \delta_N(S, 1)$
$= \{P\} \cup \{P\}$
$= \{P\}$

Step 3: Final state, $F_D = \{\{Q, S\}, \{Q\}, \{P, Q, R\}, \{Q, R\}, \{S\}, \{Q, R, S\}, \{R, S\}\}$

$\therefore$ The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

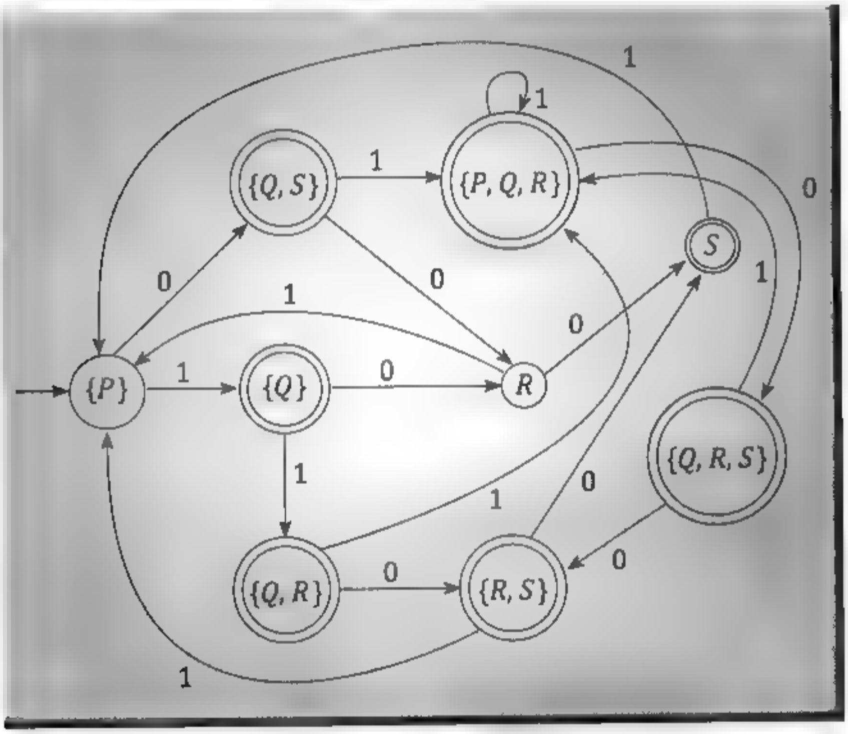
Where $Q_D = \{\{P\}, \{Q, S\}, \{Q\}, \{R\}, \{P, Q, R\}, \{Q, R\}, \{S\}, \{Q, R, S\}, \{R, S\}$

$\Sigma = \{0, 1\}$

$q_0 = \{P\}$

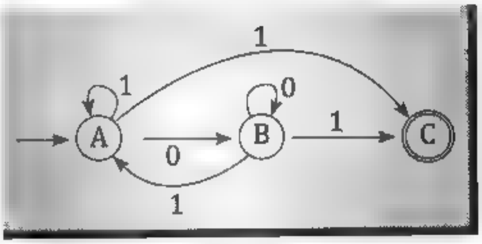
$F_D = \{\{Q, S\}, \{Q\}, \{P, Q, R\}, \{Q, R\}, \{S\}, \{Q, R, S\}, \{R, S\}\}$

δ_D	0	1
$\rightarrow \{P\}$	$\{Q, S\}$	$\{Q\}$
$* \{Q, S\}$	$\{R\}$	$\{P, Q, R\}$
$* \{Q\}$	$\{R\}$	$\{Q, R\}$
$\{R\}$	$\{S\}$	$\{P\}$
$* \{P, Q, R\}$	$\{Q, R, S\}$	$\{P, Q, R\}$
$* \{Q, R\}$	$\{R, S\}$	$\{P, Q, R\}$
$* \{S\}$	$\emptyset$	$\{P\}$
$* \{Q, R, S\}$	$\{R, S\}$	$\{P, Q, R\}$
$* \{R, S\}$	$\{S\}$	$\{P\}$



- Problem 10 -

Convert the following NFA to DFA.



Solution

The given NFA

$N = (Q_N, \Sigma, \delta, q_0, F_N)$

Where $Q_N = \{ A, B, C \}$

$\Sigma = \{ 0, 1 \}$

$q_0 = \{ A \}$

$F_N = \{ C \}$

δ_N	0	1
$\rightarrow A$	$\{ B \}$	$\{ A, C \}$
B	$\{ B \}$	$\{ A, C \}$
$* C$	$\emptyset$	$\emptyset$

The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

$\Sigma = \{ 0, 1 \}$

Step 1: $q_0 = \{ A \}$

Step 2: **δ_D from $\{ A \}$ on Σ**

$\delta_D (A, 0) = \delta_N (A, 0)$
$= \{ B \}$

$\delta_D (A, 1) = \delta_N (A, 1)$
$= \{ A, C \}$

δ_D from $\{ B \}$ on Σ

$\delta_D (B, 0) = \delta_N (B, 0)$
$= \{ B \}$

$\delta_D (B, 1) = \delta_N (B, 1)$
$= \{ A, C \}$

δ_D from $\{ A, C \}$ on Σ

$\delta_D (\{ A, C \}, 0) = \delta_N (A, 0) \cup \delta_N (C, 0)$
$= \{ B \} \cup \emptyset = \{ B \}$

$\delta_D (\{ A, C \}, 1) = \delta_N (A, 1) \cup \delta_N (C, 1)$
$= \{ A, C \} \cup \emptyset$
$= \{ A, C \}$

Step 3: Final state, $F_D = \{ \{ A, C \} \}$

$\therefore$ The equivalent DFA,

$\mathbf{D = (Q_D, \Sigma, \delta_D, q_0, F_D)}$

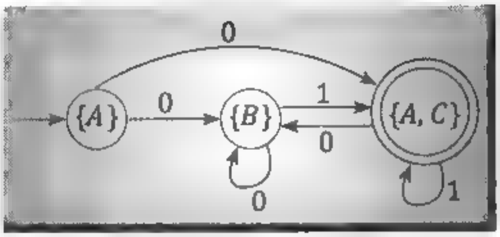
Where $Q_D = \{ \{ A \}, \{ B \}, \{ A, C \} \}$

$\Sigma = \{ 0, 1 \}$

$q_0 = \{ A \}$

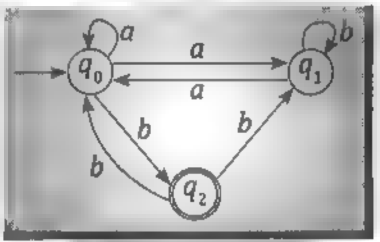
$F_D = \{ A, C \}$

δ_D	0	1
$\rightarrow \{ A \}$	$\{ B \}$	$\{ A, C \}$
$\{ B \}$	$\{ B \}$	$\{ A, C \}$
$* \{ A, C \}$	$\{ B \}$	$\{ A, C \}$



Problem 11

Construct a DFA equivalent to the following NFA



Solution

The given NFA

$$N = (Q_N, \Sigma, \delta_N, q_0, F_N)$$

Where $Q_N = \{ q_0, q_1, q_2 \}$

$\Sigma = \{ a, b \}$

$q_0 = \{ q_0 \}$

$F_N = \{ q_2 \}$

δ_N	a	b
$\rightarrow q_0$	$\{ q_0, q_1 \}$	$\{ q_2 \}$
q_1	$\{ q_0 \}$	$\{ q_1 \}$
$*q_2$	$\emptyset$	$\{ q_0, q_1 \}$

The equivalent DFA

$$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$$

$$\Sigma = \{ a, b \}$$

Step 1: $q_0 = \{ q_0 \}$

Step 2: **δ_D from $\{ q_0 \}$ on Σ**

$$\delta_D(q_0, a) = \delta_N(q_0, a) = \{ q_0, q_1 \}$$

$$\delta_D(q_0, b) = \delta_N(q_0, b) = \{ q_2 \}$$

δ_D from $\{ q_0, q_1 \}$ on Σ

$$\delta_D(\{q_0, q_1\}, a) = \delta_N(q_0, a) \cup \delta_N(q_1, a) = \{ q_0, q_1 \} \cup \{ q_0 \} = \{ q_0, q_1 \}$$

$\delta_D(\{q_0, q_1\}, b) = \delta_N(q_0, b) \cup \delta_N(q_1, b)$

$= \{q_2\} \cup \{q_1\}$

$= \{q_1, q_2\}$

δ_D from $\{q_2\}$ on Σ

$\delta_D(q_2, a) = \delta_N(q_2, a)$

$= \emptyset$

$\delta_D(q_2, b) = \delta_N(q_2, b)$

$= \{q_0, q_1\}$

δ_D from $\{q_1, q_2\}$ on Σ

$\delta_D(\{q_1, q_2\}, a) = \delta_N(q_1, a) \cup \delta_N(q_2, a)$

$= \{q_0\} \cup \emptyset$

$= \{q_0\}$

$\delta_D(\{q_1, q_2\}, b) = \delta_N(q_1, b) \cup \delta_N(q_2, b)$

$= \{q_1\} \cup \{q_0, q_1\}$

$= \{q_0, q_1\}$

Step 3: Final state, $F_D = \{\{q_2\}, \{q_1, q_2\}\}$

∴ The equivalent DFA

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

Where

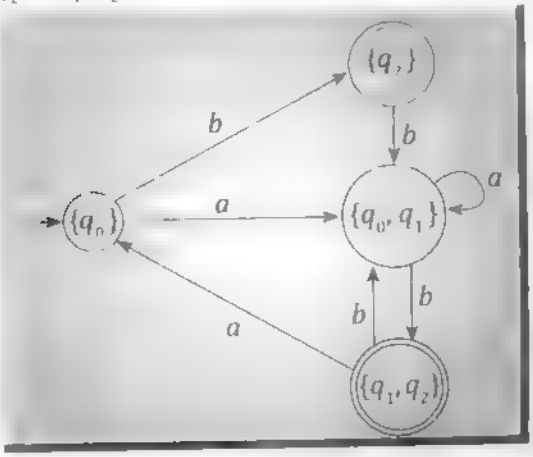
$Q_D = \{\{q_0\}, \{q_0, q_1\}, \{q_2\}, \{q_1, q_2\}\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{\{q_2\}, \{q_1, q_2\}\}$

δ_D	a	b
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_2\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$* \{q_2\}$	$\emptyset$	$\{q_0, q_1\}$
$* \{q_1, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$



1.7.3 - Theorem

If there exists NFA, N = $(Q_N, \Sigma, \delta_N, q_0, F_N)$ which accepts the language $L(N)$, there exists an equivalent DFA, D = $(Q_D, \Sigma, \delta_D, q_0, F_D)$ such that $L(D) = L(N)$

OR

Let L be a language accepted by a NFA, then there exists a DFA that accepts L.

Proof: We have to prove that $\delta_D^*(q_0, w) = \delta_N^*(q_0, w)$

We know that if Q_N represent states of NFA then power set of Q_N which contains the set of subsets of Q_N are the states of DFA denoted by Q_D. The DFA interprets each set as a single state in DFA.

Basis : Consider a string $w = \in$, where $|w| = 0$

$\delta_D^*(\{q_0\}, \in) = \{q_0\}$, by definition of extended transition function of DFA.

$\delta_N^*(\{q_0\}, \in) = \{q_0\}$, by definition of extended transition function of NFA.

Hence $\delta_D^*(q_0, w) = \delta_N^*(q_0, w)$ for some w where $|w| = n$. Now, it is required to prove that the statement. $\delta_D^*(q_0, w) = \delta_N^*(q_0, w)$ is true for some w where $|w| = n + 1$.

Inductive Proof: Let $w = x\,a$, where a is the last symbol of w and x is the remaining string of w.

$$\begin{aligned} \text{So,} \quad |x| &= n \text{ and} \\ |x\,a| &= |w| \\ &= n+1 \end{aligned}$$

By extended transition function definition of NFA, we know that

$$\begin{aligned} \delta_D^*(q_0, w) &= \delta_N^*(q_0, x\,a) \\ &= \delta_N(\delta_N^*(q_0, x), a) \qquad \dots (1) \end{aligned}$$

Now, x is the string to be processed. After consuming the string x, let the states of the machine be $\{P_i, P_j, \_\_\_\_\_ P_k\}$

i.e., $\delta_N^*(q_0, x) = \{P_i, P_j, \_\_\_\_\_ P_k\}$

Substituting this in eqn. (1)

$$\begin{aligned} \delta_N^*(q_0, w) &= \delta_N(\{P_i, P_j, \_\_\_\_\_ P_k\}, a) \\ &= \delta_N(P_i, a) \cup \delta_N(P_j, a) \cup \_\_\_\_ \delta_N(P_k, a) \qquad \dots (2) \end{aligned}$$

By extended transition function definition of DFA we know that

$$\begin{aligned} \delta_N^*(q_0, w) &= \delta_D^*(q_0, x\,a) \\ &= \delta_D^*(\delta_D^*(q_0, x), a) \qquad \dots (3) \end{aligned}$$

Now, x is the string to be processed and after consuming the string x, let the states of the machine be $\{P_i, P_j, \_\_\_\_\_ P_k\}$

i.e., $\delta_D^*(q_0, x) = \{P_i, P_j \_\_\_\_\_ P_k\}$

Substituting this in equation (3)

$$\delta^*_D (q_0, w) = \delta_D \{P_1, P_2 \_\_\_\_\_ P_k\}, a)$$
$$= \delta_N (P_1, a) \cup \delta_N (P_2, a) \cup \_\_\_\_ \delta_N (P_k, a) \qquad \dots (4)$$

By comparing equation (2) and (4)

$$\delta_D^* (\{q_0\}, w) = \delta_N^* (\{q_0\}, w)$$

so, if $\delta_D^* (\{q_0\}, w)$ is in F_N and $\delta_N^* (\{q_0\}, w)$ is in F_N,

then both enters into final state accepting the same language. Thus $L(N) = L(D)$.

Hence, the proof

1.8 ∈ - NFA

In some situations, it would be useful to enhance NFAs by allowing transitions that are not triggered by any input symbol; such transitions are called ∈ - transitions. If there is a transition from one state to another state without any input (∈). It is called ∈ - transition. An NFA with zero or more ∈ - transitions is called an ∈ - NFA.

Mathematical Representation of ∈ - NFA

Definition: ∈ - NFA is a five tuple $(Q, \Sigma, \delta, q_0, F)$

Where Q is a non-empty finite set of states.

Σ is a non-empty finite set of input symbols.

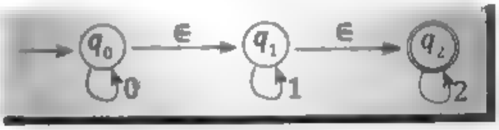
$q_0 \subseteq Q$, is the start state

$F \subseteq Q$, is the set of accepting/final states

$\delta: Q \times (\Sigma \cup \in) \longrightarrow 2^q$, is the transition function.

Based on the current state there can be a transition to other states with or without any input symbols.

Example :



∈ - NFA, $E = (Q, \Sigma, \delta, q_0, F)$

Where $Q = \{ q_0, q_1, q_2\}$

$\Sigma = \{ 0, 1, 2 \}$

$q_0 = \{ q_0 \}$

$F_D = \{q_2\}$

δ	0	1	2	∈
$\rightarrow q_0$	q_0	∅	∅	q_1
q_1	∅	q_1	∅	q_2
$*q_2$	∅	∅	q_2	∅

1.8.1 ∈ - Closure

∈ - Closure (q) is the set of all states which are reachable from state q without processing any input symbol i.e., it is just the set of states that can be reached from q on ∈ - transition only.

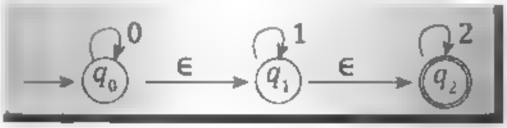
Steps to Find ∈ - Closure (q)

Step 1: q is added to ∈ - Closure (q)

Step 2: If q_1 is in ∈ - Closure (q) and there is an edge labelled ∈ from q_1 to q_2, then q_2 is added to ∈ - Closure (q), if q_2 is not already there. This is repeated until no more states can be added to ∈ - Closure (q).

Example :

Consider the following ∈ - NFA.



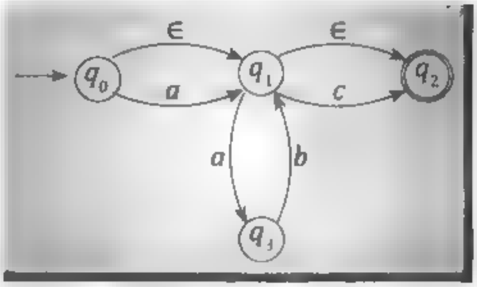
∈ - closure (q_0) = { q_0, q_1, q_2 }

∈ - closure (q_1) = { q_1, q_2 }

∈ - closure (q_2) = { q_2 }

Problem 1

Find the ∈ - closure of all states for the given ∈ - NFA.



Solution

∈ - closure (q_0) = { q_0, q_1, q_2 }

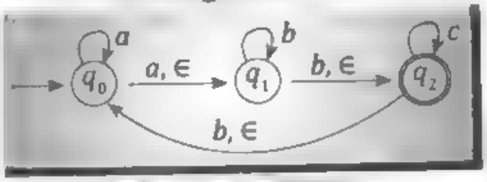
∈ - closure (q_1) = { q_1, q_2 }

∈ - closure (q_2) = { q_2 }

∈ - closure (q_3) = { q_3 }

Problem 2

Find the ϵ - closure of all states for the given ϵ - NFA.



Solution

$\in$ - closure $(q_0) = \{ q_0, q_1, q_2 \}$

$\in$ - closure $(q_1) = \{ q_1, q_2, q_0 \}$

$= \{ q_0, q_1, q_2 \}$

$\in$ - closure $(q_2) = \{ q_2, q_0, q_1 \}$

$= \{ q_0, q_1, q_2 \}$

1.8.2 - Conversion from $\in$ - NFA to DFA

Let E = $(Q_E, \sum, \delta_E, q_0, F_E)$ be an $\in$ - NFA. The equivalent DFA, D = $(Q_D, \sum, \delta_D, q_{0D}, F_D)$ can be constructed as follows.

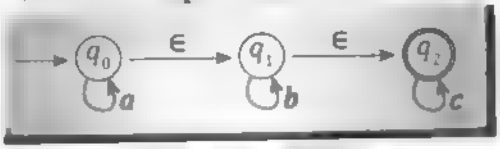
Step 1: Start state of DFA is $\in$ - closure (q_0), where q_0 is the start state of $\in$ - NFA.

Step 2: Let the $\in$ - NFA transition be δ_E $(\{q_i, q_j, ----q_k\}, a) = \{P_1, P_2, ------P_m\}$ then transition of DFA, $\delta_D(\{q_i, q_j, ----q_k\}, a) = \in$ - closure $(\{P_1, P_2, ----P_m\})$

Step 3: If $\{q_i, q_j, ----q_k\}$ is a state in DFA and if this set contains at least one final state of $\in$ - NFA, then $\{q_i, q_j, ----q_k\}$ is a final state of DFA.

Problem 3

Convert the following $\in$ - NFA to its equivalent DFA.



Solution

The given $\in$ - NFA

$E = (Q_E, \Sigma, \delta_E, q_0, F_E)$

Where

$Q_E = \{ q_0, q_1, q_2 \}$

$\Sigma = \{ a, b, c \}$

$q_0 = \{ q_0 \}$

$F_E = \{ q_2 \}$

δ	a	b	c	$\in$
$\rightarrow q_0$	q_0	$\emptyset$	$\emptyset$	q_1
q_1	$\emptyset$	q_1	$\emptyset$	q_2
$*q_2$	$\emptyset$	$\emptyset$	q_2	$\emptyset$

The equivalent DFA,

$D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$

$\Sigma = \{ a, b, c \}$

Step 1: Start state of DFA

$q_{0D} = \in\text{- closure } (q_0) = \{ q_0, q_1, q_2 \}$

Step 2: Transitions of DFA

δ_D from $\{q_0, q_1, q_2\}$ on Σ

$\delta_D (\{q_0, q_1, q_2\}, a) = \in\text{- closure } (\delta_E (\{q_0, q_1, q_2\}, a))$

$= \in\text{- closure } (\delta_E (q_0, a) \cup \delta_E (q_1, a) \cup \delta_E (q_2, a))$

$= \in\text{- closure } (q_0 \cup \emptyset \cup \emptyset)$

$= \in\text{- closure } (q_0)$

$= \{q_0, q_1, q_2\}$

$\delta_D (\{q_0, q_1, q_2\}, b) = \in\text{- closure } (\delta_E (\{q_0, q_1, q_2\}, b))$

$= \in\text{- closure } (\delta_E (q_0, b) \cup \delta_E (q_1, b) \cup \delta_E (q_2, b))$

$= \in\text{- closure } (\emptyset \cup q_1 \cup \emptyset)$

$= \in\text{- closure } (q_1)$

$= \{q_1, q_2\}$

$\delta_D (\{q_0, q_1, q_2\}, c) = \in\text{- closure } (\delta_E (\{q_0, q_1, q_2\}, c))$

$= \in\text{- closure } ((\delta_E (q_0, c) \cup \delta_E (q_1, c) \cup \delta_E (q_2, c))$

$= \in\text{- closure } (\emptyset \cup \emptyset \cup q_2)$

$= \in\text{- closure } (q_2)$

$= \{ q_2 \}$

δ_D from $\{ q_1, q_2\}$ on Σ

$\delta_D (\{ q_1, q_2 \}, a) = \in\text{- closure } (\delta_E (\{q_1, q_2\}, a))$

$= \in\text{- closure } \delta_E (q_1, a) \cup \delta_E (q_2, a)))$

$= \in\text{- closure } (\emptyset \cup \emptyset)$

$= \in\text{- closure } (\emptyset)$

$= \emptyset$

$\delta_D (\{ q_1, q_2 \}, b) = \in\text{- closure } (\delta_E (\{q_1, q_2\}, b))$

$= \in\text{- closure } (\delta_E (q_1, b) \cup \delta_E (q_2, b))$

$= \in\text{- closure } (q_1 \cup \emptyset)$

$= \in\text{- closure } (q_1)$

$= \{q_1, q_2\}$

$\delta_D(\{q_1, q_2\}, c) = \in\text{- closure }(\delta_E(\{q_1, q_2\}, c)$

$= \in\text{- closure }(\delta_E(q_1, c) \cup \delta_E(q_2, c))$

$= \in\text{- closure }(\emptyset \cup q_2)$

$= \in\text{- closure }(q_2)$

$= \{q_2\}$

δ_D from $\{q_2\}$ on Σ

$\delta_D(\{q_2\}, a) = \in\text{- closure }(\delta_E\{q_2\}, a)$

$= \in\text{- closure }(\emptyset)$

$= \emptyset$

$\delta_D(\{q_2\}, b) = \in\text{- closure }(\delta_E(\{q_2\}, b))$

$= \in\text{- closure }(\emptyset)$

$= \emptyset$

$\delta_D(\{q_2\}, c) = \in\text{- closure }(\delta_E(\{q_2\}, c))$

$= \in\text{- closure }(q_2)$

$= \{q_2\}$

Step 3: Since q_2 is the final state of $\in$ - NFA, all the sets having q_2 as a member will be the final state of DFA, $F_D = \{\{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\}\}$

$\therefore$ The equivalent DFA $\mathbf{D} = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$

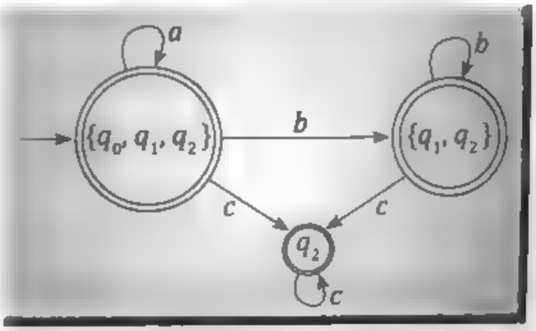
Where $Q_D = \{\{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\}\}$

$\Sigma = \{a, b, c\}$

$q_{0D} = \{q_0, q_1, q_2\}$

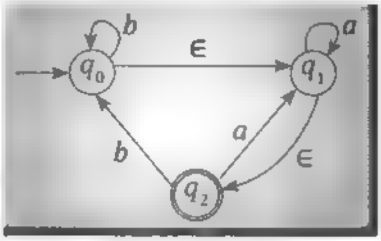
$F_D = \{\{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\}\}$

δ_D	a	b	c
$\rightarrow * \{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$* \{q_1, q_2\}$	$\emptyset$	$\{q_1, q_2\}$	$\{q_2\}$
$* \{q_2\}$	$\emptyset$	$\emptyset$	$\{q_2\}$



PROBLEM 4

Convert the following $\in$ - NFA to its equivalent DFA.



SOLUTION

The given $\in$ - NFA E = $(Q_E, \Sigma, \delta_E, q_0, F_E)$

Where

$Q_E = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$q_0 = q_0$

$F_E = \{q_2\}$

δ_E	a	b	$\in$
$\rightarrow q_0$	$\emptyset$	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*q_2$	$\{q_1\}$	$\{q_0\}$	$\emptyset$

The equivalent DFA. D = $(Q_D, \Sigma, \delta_D, q_{0D}, F_D)$, $\Sigma = \{a, b\}$

Step 1: Start state of DFA

$q_{0D} = \in$ - closure $(q_0) = \{q_0, q_1, q_2\}$

Step 2: Transitions of DFA

δ_D from $\{q_0, q_1, q_2\}$ on Σ

$\delta_D(\{q_0, q_1, q_2\}, a) = \in$ - closure $(\delta_E(\{q_0, q_1, q_2\}, a))$

$= \in$ - closure $(\delta_E(q_0, a) \cup \delta_E(q_1, a) \cup \delta_E(q_2, a))$

$= \in$ - closure $(\emptyset \cup q_1 \cup q_1)$

$= \in$ - closure (q_1)

$= \{q_1, q_2\}$

$\delta_D(\{q_0, q_1, q_2\}, b) = \in$ - closure $(\delta_E(\{q_0, q_1, q_2\}, b))$

$= \in$ - closure $(\delta_E(q_0, b) \cup \delta_E(q_1, b) \cup \delta_E(q_2, b))$

$= \in$ - closure $(q_0 \cup \emptyset \cup q_0)$

$= \in$ - closure (q_0)

$= \{q_0, q_1, q_2\}$

δ_D from $\{q_1, q_2\}$ on Σ

$\delta_D(\{q_1, q_2\}, a) = \in$ - closure $(\delta_E\{q_1, q_2\}, a)$

$= \in$ - closure $(\delta_E(q_1, a) \cup \delta_E(q_2, a))$

$= \in$ - closure $(q_1 \cup q_1)$

$= \in$ - closure (q_1)

$= \{q_1, q_2\}$

$\delta_D(\{q_1, q_2\}, b) = \in$ - closure $(\delta_E(\{q_1, q_2\}, b))$

$= \in$ - closure$(\delta_E(q_1, b) \cup \delta_E(q_2, b))$

$= \in$ - closure $(\emptyset \cup q_0)$

$= \in$ - closure (q_0)

$= \{q_0, q_1, q_2\}$

Step 3: Final state of DFA $F_D = \{\{q_0, q_1, q_2\}, \{q_1, q_2\}\}$

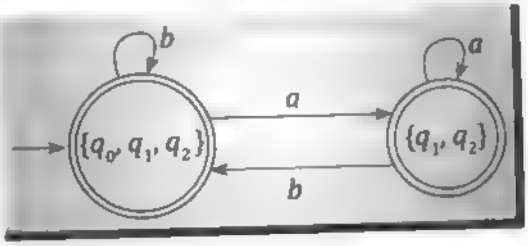
∴ The equivalent DFA **D** $= (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$

$Q_D = \{\{q_0, q_1, q_2\}, \{q_1, q_2\}\}$

$\Sigma = \{a, b\}$

$q_{0D} = \{q_0, q_1, q_2\}$

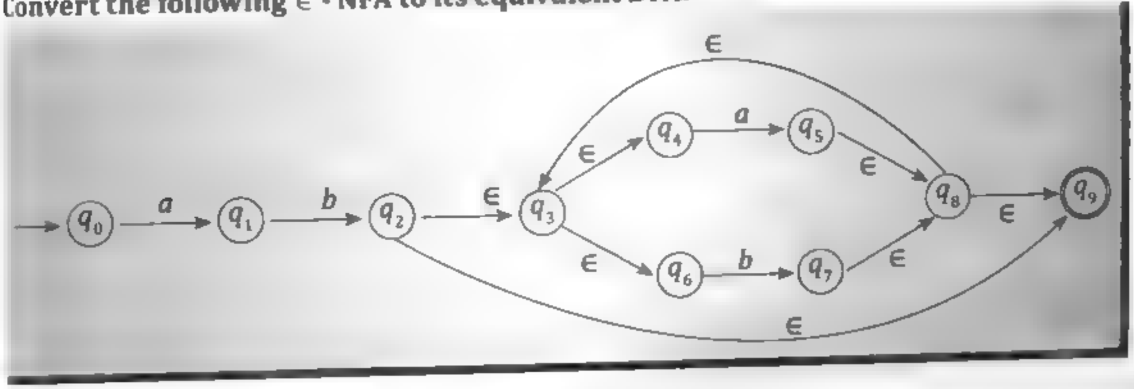
$F_D = \{\{q_0, q_1, q_2\}, \{q_1, q_2\}\}$



δ_D	a	b
$\rightarrow *\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
$*\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$

PROBLEM 5

Convert the following ∈ - NFA to its equivalent DFA.



SOLUTION

The given ∈ - NFA,

$E = (Q_E, \sum, \delta_E, q_{0D}, F_E)$

Where

$Q_E = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$

$\sum = \{a, b\}$,

$q_0 = q_0$

$F_E = \{q_7\}$

δ_E	a	b	$\in$
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\{q_2\}$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\{q_3, q_4\}$
q_3	$\emptyset$	$\emptyset$	$\{q_4, q_6\}$
q_4	$\{q_5\}$	$\emptyset$	$\emptyset$
q_5	$\emptyset$	$\emptyset$	$\{q_8\}$
q_6	$\emptyset$	$\{q_7\}$	$\emptyset$
q_7	$\emptyset$	$\emptyset$	$\{q_8\}$
q_8	$\emptyset$	$\emptyset$	$\{q_3, q_9\}$
$* q_9$	$\emptyset$	$\emptyset$	$\emptyset$

Step 1: Start state of DFA, δ_{0D} = ∈ - closure $(q_0) = \{q_0\}$

Step 2: Transitions of DFA

δ_D from $\{q_0\}$ on $\sum$

$\delta_D(q_0, a) = \in\text{- closure}(\delta_E(\{q_0\}, a)) = \in\text{- closure}(q_1) = \{q_1\}$

$\delta_D(q_0, b) = \in\text{- closure}(\delta_E(\{q_0\}, b)) = \in\text{- closure}(\emptyset) = \emptyset$

δ_D from $\{q_1\}$ on $\sum$

$\delta_D(\{q_1\}, a) = \in\text{- closure}(\delta_E(\{q_1\}, a)) = \in\text{- closure}(\emptyset) = \emptyset$

$\delta_D(\{q_1\}, b) = \in\text{- closure}(\delta_E(\{q_1\}, b)) = \in\text{- closure}(q_2) = \{q_2, q_3, q_4, q_6, q_9\}$

δ_D from $\{q_2, q_3, q_4, q_6, q_9\}$ on $\sum$

$\delta_D(\{q_2, q_3, q_4, q_6, q_9\}, a)$

$= \in\text{- closure}(\delta_E(\{q_2, q_3, q_4, q_6, q_9\}, a))$

$= \in\text{- closure}(\delta_E(q_2, a) \cup \delta_E(q_3, a) \cup \delta_E(q_4, a) \cup \delta_E(q_6, a) \cup \delta_E(q_9, a))$

$= \in\text{- closure}(\emptyset \cup \emptyset \cup \{q_5\} \cup \emptyset \cup \emptyset)$

$= \in\text{- closure}(q_5)$

$= \{q_5, q_8, q_3, q_4, q_6, q_9\}$

$= \{q_3, q_4, q_5, q_6, q_8, q_9\}$

$\delta_D(\{q_2, q_3, q_4, q_6, q_9\}, b)$

$= \in\text{- closure}(\delta_E(\{q_2, q_3, q_4, q_6, q_9\}, \text{b}))$

$= \in\text{- closure}(\delta_E(q_2, b) \cup \delta_E(q_3, b) \cup \delta_E(q_4, b) \cup \delta_E(q_6, b) \cup \delta_E(q_9, b))$

$= \in\text{- closure}(\emptyset \cup \emptyset \cup \emptyset \cup \{q_7\} \cup \emptyset)$

$= \in\text{- closure}(q_7)$

$= \{q_7, q_8, q_3, q_4, q_6, q_9\}$

$= \{q_3, q_4, q_6, q_7, q_8, q_9\}$

δ_D from $\{ q_3, q_4, q_5, q_6, q_8, q_9\}$ on Σ

$\delta_D (\{ q_3, q_4, q_5, q_6, q_8, q_9\}, a)$

$= \in$ - closure $(\delta_E (\{q_3, q_4, q_5, q_6, q_8, q_9\}, a))$

$= \in$ - closure $(\delta_E (q_3, a) \cup \delta_E (q_4, a) \cup \delta_E (q_5, a) \cup \delta_E (q_6, a) \cup \delta_E (q_8, a) \cup \delta_E (q_9, a))$

$= \in$ - closure $(\emptyset \cup \{ q_5 \} \cup \emptyset \cup \emptyset \cup \emptyset \cup \emptyset \cup \emptyset)$

$= \in$ - closure (q_5)

$= \{ q_5, q_8, q_3, q_4, q_6, q_9 \}$

$= \{ q_3, q_4, q_5, q_6, q_8, q_9 \}$

$\delta_D (\{ q_3, q_4, q_5, q_6, q_8, q_9\}, b)$

$= \in$ - closure $(\delta_E (\{q_3, q_4, q_5, q_6, q_8, q_9\}, b))$

$= \in$ - closure $(\delta_E (q_3, b) \cup \delta_E (q_4, b) \cup \delta_E (q_5, b) \cup \delta_E (q_6, b) \cup \delta_E (q_8, b) \cup \delta_E (q_9, b))$

$= \in$ - closure $(\emptyset \cup \emptyset \cup \emptyset \cup \{ q_7 \} \cup \emptyset \cup \emptyset)$

$= \in$ - closure (q_7)

$= \{ q_7, q_8, q_3, q_4, q_6, q_9 \}$

$= \{ q_3, q_4, q_6, q_7, q_8, q_9 \}$

δ_D from $\{ q_3, q_4, q_6, q_7, q_8, q_9\}$ on Σ

$\delta_D (\{ q_3, q_4, q_6, q_7, q_8, q_9\}, a)$

$= \in$ - closure $(\delta_E (q_3, a) \cup \delta_E (q_4, a) \cup \delta_E (q_6, a) \cup \delta_E (q_7, a) \cup \delta_E (q_8, a) \cup \delta_E (q_9, a))$

$= \in$ - closure $(\emptyset \cup \{ q_5 \} \cup \emptyset \cup \emptyset \cup \emptyset \cup \emptyset)$

$= \in$ - closure (q_5)

$= \{ q_5, q_8, q_3, q_4, q_6, q_9 \}$

$= \{ q_3, q_4, q_5, q_6, q_8, q_9 \}$

$\delta_D (\{ q_3, q_4, q_6, q_7, q_8, q_9\}, b)$

$= \in$ - closure $(\delta_E (q_3, b) \cup \delta_E (q_4, b) \cup \delta_E (q_6, b) \cup \delta_E (q_7, b) \cup \delta_E (q_8, b) \cup \delta_E (q_9, b))$

$= \in$ - closure $(\emptyset \cup \emptyset \cup \{ q_7 \} \cup \emptyset \cup \emptyset \cup \emptyset)$

$= \in$ - closure (q_7)

$= \{ q_7, q_8, q_3, q_4, q_6, q_9 \}$

$= \{ q_3, q_4, q_6, q_7, q_8, q_9 \}$

Step 3: The final state $F_D = \{ \{ q_2, q_3, q_4, q_6, q_9\}, \{ q_3, q_4, q_5, q_6, q_8, q_9 \}, \{ q_3, q_4, q_6, q_7, q_8, q_9 \} \}$

$\therefore$ the equivalent DFA $\mathbf{D = (Q_D, \Sigma, \delta_D, q_0, F_D)}$

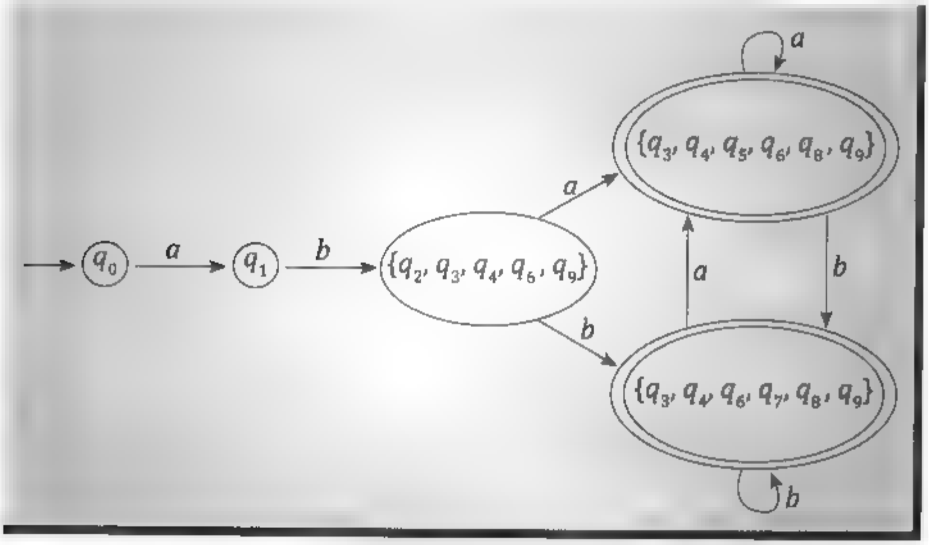
Where $Q_D = \{ \{ q_0\}, \{ q_1\}, \{ q_2, q_3, q_4, q_6, q_9\}, \{ q_3, q_4, q_5, q_6, q_8, q_9\}, \{ q_3, q_4, q_6, q_7, q_8, q_9\} \}$

$\Sigma = \{ a, b \}$

$q_{0D} = \{ q_0\}$

$F_D = \{ \{ q_2, q_3, q_4, q_6, q_9\}, \{ q_3, q_4, q_5, q_6, q_8, q_9\}, \{ q_3, q_4, q_6, q_7, q_8, q_9\} \}$

δ_D	a	b
$\rightarrow \{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\emptyset$	$\{q_2, q_3, q_4, q_6, q_9\}$
$*\{q_2, q_3, q_4, q_6, q_9\}$	$\{q_3, q_4, q_5, q_6, q_8, q_9\}$	$\{q_3, q_4, q_6, q_7, q_8, q_9\}$
$*\{q_3, q_4, q_5, q_6, q_8, q_9\}$	$\{q_3, q_4, q_5, q_6, q_8, q_9\}$	$\{q_3, q_4, q_6, q_7, q_8, q_9\}$
$*\{q_3, q_4, q_6, q_7, q_8, q_9\}$	$\{q_3, q_4, q_5, q_6, q_8, q_9\}$	$\{q_3, q_4, q_6, q_7, q_8, q_9\}$


q_0
a
q_1
b
$\{q_2, q_3, q_4, q_6, q_9\}$
a
b
$\{q_3, q_4, q_5, q_6, q_8, q_9\}$
a
a
b
$\{q_3, q_4, q_6, q_7, q_8, q_9\}$
b

1.8 THEOREM

A language L is accepted by DFA, D, if and only if L is accepted by an $\in$ - NFA, E.

Proof: Let E = $(Q_E, \Sigma, \delta_E, q_0, F_E)$ be an $\in$ - NFA and let the DFA obtained from it be D = $(Q_D, \Sigma, \delta_D, q_{0D}, F_D)$.

Since all the transition of DFA are obtained from the transitions of $\in$ - NFA, the language accepted by $\in$ - NFA is also accepted by DFA. This can be proved using mathematical induction as shown below:

Basis : If $w = \in$, we know that $\delta_E^* (q_0, \in) = \in$ - closure (q_0) - - - - - (1)

We also know that $q_{0D} = \in$ - closure (q_0) is correct since that is how the start state of DFA is obtained so, if $w = \in$, then $\delta_D^* (q_{0D}, \in) = \in$ - closure (q_0) - - - - - (2)

From eqn. (1) and (2)

$\delta_E^* (q_{0D}, \in) = \delta_D^* (q_{0D}, \in)$

Thus, we have proved that language accepted by $\in$ - NFA is same as language accepted by DFA when $w = \in$.

Introduction Hypothesis: Let $w = x$ and assume that the result is true for the string x

i.e., $\delta_E^*(q_0, x) = \delta_D^*(q_{0D}, x) = \{q_1, q_2, \text{-----} q_k\}$

is true for the string x.

Now we have to prove that the result is true for $w = x\,a$. By definition of δ^* for $\in$ - NFA, we compute $\delta_E^*(q_0, w)$ as shown below:

Let $\delta_E(\{q_1, q_1, \text{---} q_k\}, a) = \{P_1, P_2, \text{------} P_m\}$ - - - - - (3)

then $\delta_E^*(q_0\, w) = \in$ - closure $(\{P_1, P_2, \text{------} P_m\})$ - - - - - (4)

Thus, $\delta_D^*(q_0, w) = \in$ - closure $(\{P_1, P_2, \text{------} P_m\})$ - - - - - (5)

Equation (3) and (4) are used in the construction of DFA from $\in$ - NFA.

Thus, $\delta_D^*(q_0, w) = \delta_E^*(q_0, w)$

1.9 COMPARISON OF DFA, NFA AND $\in$ - NFA

Difference between DFA, NFA and $\in$ - NFA

DFA	NFA	$\in$ - NFA
DFA is a 5 - tuple $D = (Q, \Sigma, \delta, q_0, F)$ $\delta\; Q \times \Sigma \rightarrow Q$	NFA is a 5 - tuple $N = (Q, \Sigma, \delta, q_0, F)$ $\delta : Q \times \Sigma \rightarrow 2^Q$	$\in$ -NFA is a 5 - tuple $E = (Q, \Sigma, \delta, q_0, F)$ $\delta : Q \times (\Sigma \cup \in) \rightarrow 2^Q$
DFA can have only one transition from a state on an input symbol	NFA can have zero, one or more transitions from a state on an input symbol	$\in$ - NFA is a NFA with zero or more $\in$ - transitions
Difficult to construct	Easy to construct	Easy to construct
Less powerful since at any point of time it will be in only one state	More powerful than DFA since at any point of time it will be in more than one state	More powerful than NFA since at any point of time it will be in more than one state with or without giving any input

REVIEW QUESTIONS

Short Answer Questions

1. Define Symbol, with an example.
2. Define Alphabet with an example.
3. Explain string, length of a string and empty string.
4. Define Language with an example.
5. Define DFA.
6. Define NFA.
7. List out the different types of finite automa.
8. Explain Language acceptance of a DFA.
9. Explain $\in$ - closure (q)

Long Answer Questions

1. Explaın Finıte automata ın detaın wıth an example
2. Wrıte short notes an Applicatıons of finite automa
3. Defıne Language and operatıons performed on the language.
4. Define extended transition function of DFA.
5. Differentiate between DFA, NFA and ∈ - NFA.
6. Define a DFA such that L(E) = L(D)
7. Construct a DFA to accept all strıngs on { *a*, *b* } having the prefix '*baa*'. Sımulate the behavio of DFA on *baabab*, *abaab*.

●✱●✱●

UNIT 2

REGULAR EXPRESSION

CHAPTER OUTLINE

2.0 Introduction

In this chapter, we define regular expressions as a means of representing certain subse of strings over Σ and will prove that regular sets are precisely those accepted by fin automata. We use pumping lemma for regular sets to prove that certain sets are not regu Then we discuss about the properties of regular expressions.

2.1 Regular Expressions (R.E)

The language accepted by finite automata is called regular language. A regular langua can be described using regular expressions, consisting of alphabets in Σ and the operato '*', '·', '+'. The order of evaluation of regular expression is determined by parenthesis an the operator precedence '*', '·' and '+' respectively.

Definition : Regular Expression

A regular expression is recursively defined as follows.

- '∅' is a regular expression denoting an empty language.
- '∈' is a regular expression denoting the language containing an empty string.
- '*a*' is a regular expression denoting the language containing only { *a* }
- If *R* is a regular expression denoting the language L_R and *S* is a regular expressio denoting the language L_S, then
  - R + S is a regular expression corresponding to the language $L_R \cup L_S$.
  - R·S is a regular expression corresponding to the language $L_R \cdot L_S$.
  - R* is a regular expression corresponding to the language L_R.
- The expressions obtained by applying any of these rules are regular expressions.

Note Any set represented by a regular expression is called regular set

Example :

Let Σ = { *a* } and regular expression, *R* = *a** then *R* is a set denoted by
L = {∈, *a*, *aa*, *aaa*, *aaaa*, _ _ _ _ _}

2.2 Basic Operations of Regular Expressions

The basic operations of regular expressions are concatenation, Union and Kleen closure

Let *R* and *S* be any two regular expressions.

Concatenation: Concatenation of regular expression R and S is $RS = \{ xy | x \in R \text{ and } y \in R\}$

Example :

If *R* = {*ab*, *c*} and *S* = {*d*, *ef*} then *RS* = { *abd*, *cd*, *abef*, *cef*}

Union: Union of regular expressions R and S is $R + S = \{x \mid x \in R \text{ or } x \in S\}$

Example:

If $R = \{ab, c\}$ and $S = \{d, ef\}$ then $R + S = \{ab, c, d, ef\}$

Kleene Closure: Kleen closure is the set of all strings, that can be made by concatenating zero or more strings in R.

Example:

If $R = \{ab, c\}$ then $R^* = \{\in, ab, c, abab, cc, abc, cab, \ldots\}$

2.3 Properties of regular expressions

Let P, Q, and R be any arbitrary regular expression. Then, the following properties are true.

1. $\in R = R \in = R$
2. $\in^* = \in$
3. $(\emptyset)^* = \in$
4. $\emptyset R = R \emptyset = \emptyset$
5. $\emptyset + R = R$
6. $R + R = R$
7. $R R^* = R^* R = R^*$
8. $(R^*)^* = R^*$
9. $\in + R R^* = R^*$
10. $(P + Q) R = PR + QR$
11. $(P + Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
12. $R^* (R^* + R) = (R^* + R) R^* = R^*$
13. $(R + R^*)^* = R^*$
14. $\in + R^* = R^*$
15. $(PQ)^* P = P (QP)^*$
16. $R^* R + R = R^* R$
17. $(PQ)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
18. $R^* R^* = (R^*)^* = R^*$

2.4 Describing regular expressions

Regular expressions	Meaning
0^*	Set of strings of zeros of any length including $\in$
0^+	Set of strings of zeros of any length excluding $\in$ (atleast one 0)
$0 + 1$	Set of strings consisting of either one 0 or one 1.
$(0 + 1)^*$	Set of strings of 0's and 1's of any length including $\in$
$(0 + 1)^+$	Set of strings of 0's and 1's of any length excluding $\in$
$(0 + 1)^* 011$	Set of strings of 0's and 1's ending with 011.
$01 (0 + 1)^*$	Set of strings of 0's and 1's starting with 01
$0^* 1^* 2^*$	Set of strings of zero or more 0's followed by zero or more 1's followed by zero or more 2's including $\in$
$0^+ 1^+ 2^+$	Set of strings of any number of 0's followed by any number of 1's followed by any number of 2's excluding $\in$
$00^* 11^* 22^*$	Set of strings of 0's, 1's and 2's with atleast one zero, followed by atleast one 1, followed by atleast one 2
$(0 + 1)^* (0 + 11)$	Set of strings of 0's and 1's ending with either 0 or 11.

$(00)^* (11)^* 1$	Set of strings consisting of even number of 0's followed by odd numbe of 1's
$(0 + 1)^* 000$	Set of strings of 0's and 1's ending with three consecutive 0's
$(11)^*$	Set of strings consisting of even number of 1's
$1 + (01)^*$	The language consists of a string 1 or strings of (01)'s that repeats zer or more times
$0 (1^* + 1)$	Set of strings of 0 followed by any number of 1's
$(1 + \epsilon) (00^* 1)^* 0^*$	Strings of 0's and 1's without any consecutive 1's
$(0 + 10)^* 1^*$	Strings of 0's and 1's ending with any number of 1's
$(0 + 1) (0 + 1)$	Strings of 0's and 1's whose length is 2
$(0 + 1)^* 00 (0 + 1)^*$	Set of strings of 0's and 1's of any length having a substring 00.
$0^* 1 (0^* 10^* 1)^* 0^*$	Set of strings starts with 0 or 1 and has atleast one 1, after the first 1 all 1's appear in pairs. Any number of 0's can appear in any place in the string. The string contains an odd number of 1's.

PROBLEM 1

Write the RE for the language accepting all combination of a's over the set, Σ = { a }

SOLUTION

$L = \{\in, a, aa, aaa, \_\_\_\_\} \therefore R = a^*$

PROBLEM 2

Design the RE for the language accepting all combinations of a's except the null strin over Σ = { a }

SOLUTION

$L = \{a, aa, aaa, aaaa, \_\_\_\_\} \therefore R = a^+$

PROBLEM 3

Design a RE for the language containing any number of a's and b's.

SOLUTION

$\Sigma = \{ a, b \}$

$L = \{\in, a, b, ab, ba, aa, bb, abab, bab, \_\_\_\_\}$

$\therefore R = (a + b)^*$

Problem 4

Construct the RE for the language containing all the strings which ends with 00 over the set $\Sigma = \{ 0, 1 \}$

Solution

$L = \{ 00, 000, 100, 0100, 1000, \_\_\_\_\_ \}$

$\therefore R = (0 + 1)^* 00$

Problem 5

Construct the RE for the language consisting of the set of all strings of 0's and 1's beginning with 0 and ending with 1.

Solution

$\Sigma = \{ 0, 1 \}$

$L = \{ 01, 001, 011, 0011, 0101, 001011, \_\_\_\_\_ \}$

$\therefore R = 0 (0 + 1)^* 1$

Problem 6

Construct a RE for the following language.

$L = \{\epsilon, 11, 1111, 111111, \_\_\_\_\}$

Solution

$\Sigma = \{ 1 \}$

Any element of L is either $\in$ or a string of even number of 1's.

$\therefore R = (11)^*$

Problem 7

Construct a RE for the language consisting of all strings of *a*'s and *b*'s beginning with a and ending with *ab*.

Solution

$\Sigma = \{ a, b \}$

$L = \{ aab, aaab, abab, aabab, aaaab, abbab, \_\_\_\_\_ \}$

$\therefore R = a (a + b)^* ab$

Problem 8

Construct a RE for the set of strings of 0's and 1's that should contain the substring 000.

Solution

$\Sigma = \{0, 1\}$

$L = \{000, 0000, 1000, 00001, 10000, 1100000, 01000, , \_\_\_\_\_\}$

$\therefore R = (0 + 1)^* 000 (0 + 1)^*$

Problem 9

Give a RE for representing the set L of strings in which every 0 is immediately followed by atleast two 1's.

Solution

$\Sigma = \{0, 1\}$

The elements of L is zero or more occurrences of 1 or 0 followed by 11.

i.e., $L = \{1, 11, 011, 1011, \_\_\_\_\}$

$\therefore R = (1 + 011)^*$

Problem 10

Construct a RE for the set of strings of *a*'s and *b*'s, ending with either *a* or *bb*.

Solution

$\Sigma = \{0, 1\}$, $L = \{a, bb, aa, abb, baa \ \_\_\_\_\}$

$\therefore R = (a + b)^* (a + bb)$

Problem 11

Construct a RE for the language, 0 followed by zero or 1's or 1 followed by zero or more 0's.

Solution

$\Sigma = \{0, 1\}$

$L = \{0, 1, 01, 10, 011, 100, \_\_\_\_\}$

$\therefore R = 01^* + 10^*$

Problem 12

Write RE for the following language, 1 followed by zero or more 0's and ending with 1.

Solution

= { 0, 1 }

L = {11, 101, 10001, ____}   ∴ $R = 10^* 1$

Problem 13

Write RE for the set of strings over alphabet {*a*, *b*, *c*} containing at least one *a* and one *b*.

Solution

$\Sigma = \{ a, b, c \}$, L = {*ab*, *aab*, *aabb*, *abbbc*, ____}

∴ $R = (a + b + c)^* ab (a + b + c)^*$

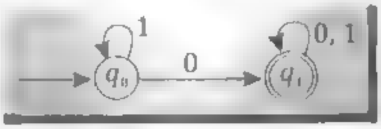
Problem 14

Draw the DFA for all strings that have atleast one 0 over the alphabet {0, 1} and write the RE.

Solution

$\Sigma = \{ 0, 1 \}$

DFA



∴ $R = 1^* 0 (0 + 1)^*$

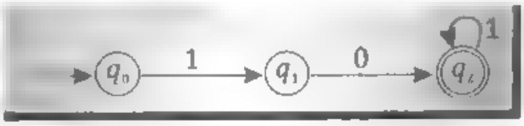
Problem 15

Draw the DFA for the set of all strings which starts from 10 and ends with any number of 1's.

Solution

$\Sigma = \{ 0, 1 \}$

DFA



. $R = 101^*$

PROBLEM 16

Obtain a RE representing strings of *a*'s and *b*'s having even length.

SOLUTION

$\Sigma = \{ a, b \}$

$L = \{ aa, ab, ba, bb, aabb, abab, \_\_\_\_ \}$

$\therefore R = ((a + b)(a + b))^*$

PROBLEM 17

Write a RE representing strings of 0's and 1's having odd length.

SOLUTION

$\Sigma = \{ 0, 1 \}$

$L = \{ 0, 1, 001, 10101, \_\_\_\_ \}$

$\therefore R = (0 + 1)((0 + 1)(0 + 1))^*$

PROBLEM 18

Obtain *a* RE for the set of strings containing exactly one a over $\Sigma = \{ a, b, c \}$

SOLUTION

$L = \{ a, ba, cab, bcabc, \_\_\_\_ \}$

$\therefore R = (b + c)^* a (b + c)^*$

PROBLEM 19

Write a RE for the set of all strings with no two consecutive *a*'s, where $\Sigma = \{ a, b \}$.

SOLUTION

$L = \{ a, aba, bab, bababa, \_\_\_\_ \}$

$\therefore R = (b + ab)^* (a + \in)$

PROBLEM 20

Obtain a RE for the set of all strings with exactly two *b*'s over $\Sigma = \{ a, b \}$

SOLUTION

$L = \{ bb, abba, ababa, \_\_\_\_ \}$

$\therefore R = a^* ba^* ba^*$

Problem 21

Obtain a RE for the set of all strings with atleast two b's over $\Sigma = \{ a, b \}$

Solution

$L = \{ bb, abab, bbbb, \_\_\_\_ \}$

$\therefore R = (a + b)^* b (a + b)^* b (a + b)^*$

Problem 22

Construct a RE defining the language consisting of all words that contain exactly 3 b's in total. Where $\sum = \{a,b\}$

Solution

L = $\{bbb, abbb, bbba, babb, bbab, bababaa, \ldots\ldots\}$

$R = a^* ba^* ba^* ba^*$

Problem 23

Obtain a RE for the set of all strings that do not end with 01, over {0,1}

Solution

It is given that the string should not end with 01. So other strings whose length is 2 and that do not end with 01 are 00, 10 and 11, 000, 010 etc., These strings can be preceded by any combinations of 0's and 1's denoted by $(0+1)^*$. These strings can end with 00 or 10 or 11.

$\therefore$ RE, R = $(0 + 1)^* (00 + 10 + 11)$

2.5 - Construction of DFA from regular expression

This procedure can be explained easily with an example.

Problem 1

Construct DFA for the regular expression $(a \mid b)^* a\, b\, b$

Solution

Step 1: Augment the given regular expression with the symbol #.

$(a \mid b)^* a\, b\, b\, \#$

Step 2: Give position to each symbol in the regular expression including # symbol.

```
(a | b)*  a   b   b   #
 1   2    3   4   5   6
```

Step 3: Find Firstpos of the given regular expression. Firstpos is a set that contains the positions of all the symbols which can be at the beginning of a valid word of the regular expression.

Firstpos becomes the start state of DFA.

Firstpos ($(a \mid b)^* abb$ #) is { 1, 2, 3 }

'1' is included in the Firstpos because of word

$\begin{matrix} a & b & a & b & b & \# \\ 1 & 2 & 3 & 4 & 5 & 6 \end{matrix}$ (or) $\begin{matrix} a & a & a & b & b & \# \\ 1 & 1 & 3 & 4 & 5 & 6 \end{matrix}$ etc....

'2' is included in the Firstpos because of the word

$\begin{matrix} b & a & b & b & \# \\ 2 & 3 & 4 & 5 & 6 \end{matrix}$ (or) $\begin{matrix} b & b & a & b & b & \# \\ 2 & 2 & 3 & 4 & 5 & 6 \end{matrix}$ (or) $\begin{matrix} b & a & a & b & b & \# \\ 2 & 1 & 3 & 4 & 5 & 6 \end{matrix}$

'3' is included in First pos because of the word

$\begin{matrix} a & b & b & \# \\ 3 & 4 & 5 & 6 \end{matrix}$

Start state of DFA q_0 = $\{ \underset{a,}{1}, \underset{b,}{2}, \underset{a}{3} \}$

Step 4: Find followpos of each symbol. Followpos of a symbol is a set which contains the positions of all the symbols which can follow the current symbol.

In the regular expression

$\begin{matrix} (a|b)^* & a & a & b & \# \\ 1\;2 & 3 & 4 & 5 & 6 \end{matrix}$

Followpos (1) = { 1, 2, 3 }

Followpos (2) = { 1, 2, 3 }

Followpos (3) = { 4 }

Followpos (4) = { 5 }

Followpos (5) = { 6 }

Followpos (6) = ∅

Step 5: Find set of states of DFA and transitions.

For the start state q_0, Find transition for Σ = {a, b}

Since a occurs in first and third position in q_0,

$$\begin{aligned} \delta(q_0, a) &= \text{Followpos}(1) \cup \text{Followpos}(3) \\ &= \{1, 2, 3\} \cup \{4\} \\ &= \{1, 2, 3, 4\} \rightarrow \textcircled{q_1} \\ &\quad \{a, b, a, b\} \end{aligned}$$

Since b occurs in second position in q_0,

$$\begin{aligned} \delta(q_0, b) &= \text{Followpos}(2) \\ &= \{1, 2, 3\} \rightarrow \textcircled{q_0} \end{aligned}$$

For the new state q_1 find the transition for $\Sigma = \{ a, b \}$

$\delta(q_1, a)$ = Followpos (1) $\cup$ Followpos (3)

= $\{1, 2, 3\} \cup \{4\}$

= $\{1, 2, 3, 4\} \rightarrow q_1$

$\delta(q_1, b)$ = Followpos (2) $\cup$ Followpos (4)

= $\{1, 2, 3\} \cup \{5\}$

= $\{1, 2, 3, 5\} \rightarrow q_2$

a b a b

For the state q_2 find the transitions.

$\delta(q_2, a)$ = Followpos (1) $\cup$ Followpos (3)

= $\{1, 2, 3\} \cup \{4\}$

= $\{1, 2, 3, 4\} \rightarrow q_1$

$\delta(q_2, b)$ = Followpos (2) $\cup$ Followpos (5)

= $\{1, 2, 3\} \cup \{6\}$

= $\{1, 2, 3, 6\} \rightarrow q_3$

a b a #

For the state q_3 find the transitions.

$\delta(q_3, a)$ = Followpos (1) $\cup$ Followpos (3)

= $\{1, 2, 3\} \cup \{4\}$

= $\{1, 2, 3, 4\} \rightarrow q_1$

$\delta(q_3, b)$ = Followpos (2)

= $\{1, 2, 3\} \rightarrow q_0$

Since there is no. new state, stop the iteration.

Step 6: The DFA M is defined as

M = $(Q, \Sigma, \delta, q_0, F)$

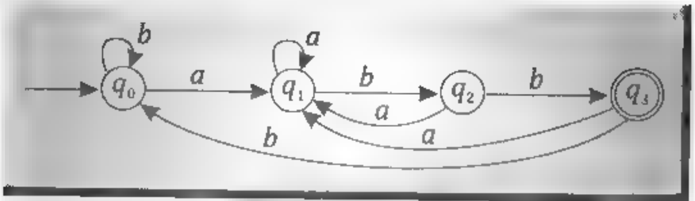
Where Q = $\{q_0, q_1, q_2, q_3\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

F = $\{q_3\}$ since # occurs in q_3

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
$*q_3$	q_1	q_0



Problem 2

Construct DFA for the regular expression $a\,b^* + b$

Solution

Step 1: Argument the regular expression with the symbol #

$(a\,b^* + b)$ #

Step 2: Give position to each symbol including # symbol

$$\underset{1}{(a}\ \ \underset{2}{b^*}\ +\ \underset{3}{b)}\ \ \underset{4}{\#}$$

Step 3: Find firstpos of the given regular expression.

Firstpos becomes the start state of DFA.

Firstpos ($(a\,b^* + b)$ #) is { 1, 3 }

Start state of DFA q_0 = { 1, 3, }

$a\ \ b$

Step 4: Find followpos of each symbol. In the regular expression

$$\underset{1}{(a}\ \ \underset{2}{b^*} + \underset{3}{b)}\,\underset{4}{\#}$$

Followpos (1) = { 2, 4 }

Followpos (2) = { 2, 4 }

Followpos (3) = { 4 }

Followpos (4) = ∅

Step 5: Find set of states of DFA and transitions

For Start state q_0 find the transitions

Since a occurs at first position in q_0

$\delta(q_0, a)$ = Followpos (1)

= { 2, 4 } → $\boxed{q_1}$

$b, \#$

$\delta(q_0, b)$ = Followpos (3)

= { 4 } → q_2

For the state q_1 find the transion

$\delta(q_1, a)$ = ∅

$\delta(q_1, b)$ = Followpos (2) ∪ Followpos (4) = { 2, 4 } ∪ { ∅ } = { 2, 4 } → q_1

For the state q_2, Find the transition

$\delta(q_2, a) = \emptyset$

$\delta(q_2, b) = \emptyset$

Step 6: The DFA M is defined as

$M = (Q, \Sigma, \delta, q_0, F)$

Where

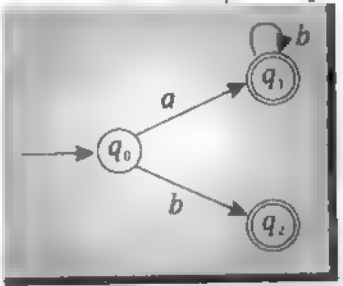
$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_1, q_2\}$ as # occurs in both q_1 and q_2.

δ	a	b
$\rightarrow q_0$	q_1	q_2
$*q_1$	∅	q_1
$*q_2$	∅	∅



2.6 CONSTRUCTION OF ∈ - NFA FROM REGULAR EXPRESSION

2.6.1 Theorem

Let 'r' be a regular expression. Then there exists an ∈ - NFA that accepts $L(r)$.

(or)

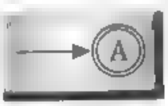
Prove that there exists a finite automation $M = (Q, \Sigma, \delta, q_0, F)$ to accept the language $L(r)$ corresponding to the regular expression r.

Proof: We show by induction on the number of operators in the regular expression 'r' that there is an NFA, M with ∈ - transitions, having one final state and no transitions out of this final state, such that $L(M) = L(r)$.

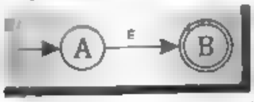
Basis of Induction (zero operators)

Consider the case where 'r' has zero operators. Then 'r' must be either ∈, ∅, or a (for some a in Σ). For each of these 3 cases, the corresponding NFA's are given as follows.

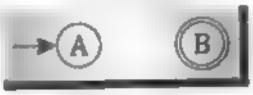
If r = ∈, then



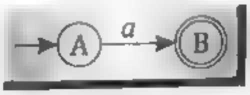
or



If r = ∅, then



If $r = a$, then



Induction Hypothesis (one or more operators)

Let us assume that the theorem is true for all regular expressions consisting of less tha i operators. ($i \geq 1$). That is if a regular expression contains less than i operators then we can construct an equivalent NFA with $\in$ - moves.

Now, we have to prove that if the RE contains i operators also, we can construct a equivalent NFA with $\in$ - moves.

Suppose r has i operators, the following 3 cases occur

Case 1: $r = r_1 + r_2$ (union)

Case 2: $r = r_1 \cdot r_2$ (concatenation)

Case 3: $r = r_1{}^*$ (closure)

Where r_1 and r_2 have less than i operators and r has i operators.

Case 1: $r = r_1 + r_2$

By induction hypothesis there exists an NFA with $\in$ - moves corresponding to r_1 and r_2.

Consider the NFA corresponding to r_1 as $M_1 = (q_1, \Sigma_1, \delta_1, q_1, f_1)$ and correspondin to r_2 as $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, f_2)$ such that $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$

Now let us construct an NFA M corresponding to r as follows.

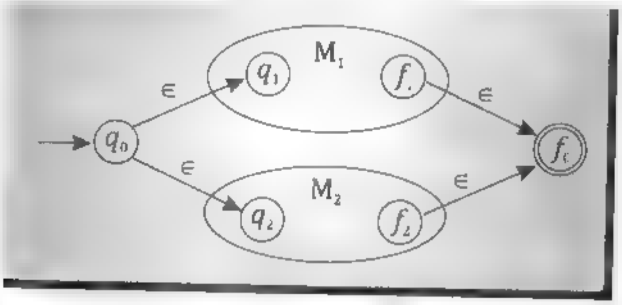
Let q_0 be a new initial state and f_0 be a new final state construct

$$M = (Q_1 \cup Q_2 \cup \{ q_0, f_0 \}, \Sigma_1 \cup \Sigma_2, \delta, q_0, f_0)$$

Where δ is defined as follows.

1. $\delta(q_0, \in) = \{ q_1, q_2)$
2. $\delta(q, a) = \delta_1(q, a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\in\}$
3. $\delta(q, a) = \delta_2(q, a)$ for q in $Q_2 - \{f_2\}$ and a in $\Sigma_2 \cup \{\in\}$
4. $\delta(f_1, \in) = f_0$
5. $\delta(f_2, \in) = f_0$

The equivalent transition diagram is as follows.



r_1 is accepted by M through the path $q_0\, q_1 \_\_\ f_1\, f_0$ and r_2 is accepted by M through the path $q_0\, q_2 \_\_\_\_ f_2\, f_0$.

Thus M accepts $r_1 + r_2$

$\therefore$ M accepts r.

Case 2: $r = r_1 \cdot r_2$

Let M_1 and M_2 be as in case 1. Now, we will construct the $\in$ - NFA, M corresponding to r as follows.

$$M = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1, f_2)$$

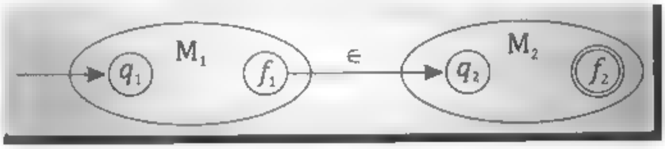
where q_1 is the initial state and f_2 is the final state. δ is defined as follows.

$\delta(f_1, \in) = q_2$

$\delta(q, a) = \delta_1(q, a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\in\}$

$\delta(q, a) = \delta_2(q, a)$ for q in Q_2 and a in $\Sigma_2 \cup \{\in\}$

The equivalent transition diagram is as follows.



M accepts r_1 from q_1 to f_1 and then using $\in$ – transition it goes to q_2

r_2 is accepted from q_2 to f_2.

$\therefore$ M accepts $r_1 \cdot r_2$

Case 3: $r = r_1^*$

By inductive hypothesis there is an NFA with $\in$ – moves corresponding to r_1 as

$$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, f_1) \text{ such that } L(M_1) = L(r_1)$$

Now, we construct an NFA, M corresponding to r as follows.

$$M = (Q_1, \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, f_0)$$

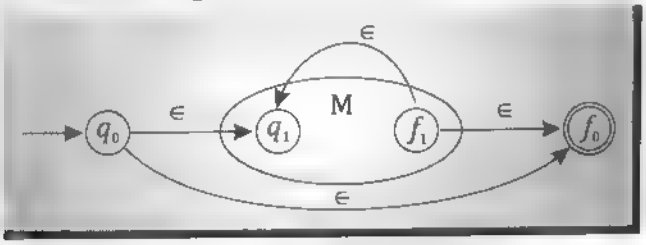
Where q_0 is the new initial state and f_0 is the new final state. δ is defined as follows.

$\delta(q_0, \in) = \{q_1, f_0\}$

$\delta(f_1, \in) = \{q_1, f_0\}$

$\delta(q_1, a) = \delta_1(q, a)$ for q in $Q_1 - \{f_1\}$ and a in $\Sigma_1 \cup \{\in\}$

The equivalent transition diagram is as follows.



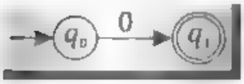
M accepts r_1^* and hence we have constructed $\in$ – NFA for r which has i operators. Hence the proof by induction.

Problem 1

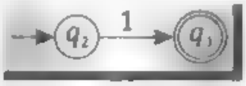
Construct an NFA with $\in$ - moves for 01.

Solution

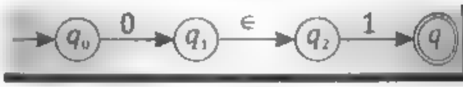
To accept 0



To accept 1



To accept 01

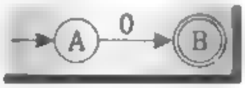


Problem 2

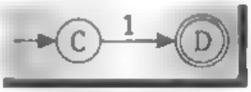
Construct on NFA with $\in$ - moves for 0 + 1.

Solution

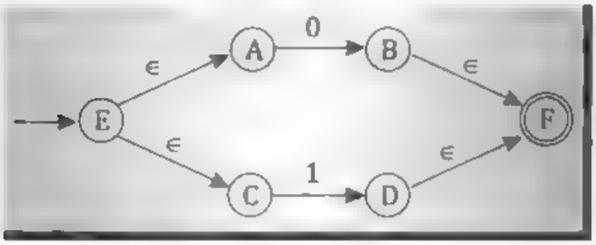
To accept 0



To accept 1



To accept 0 + 1

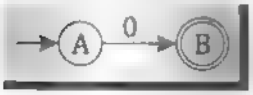


Problem 3

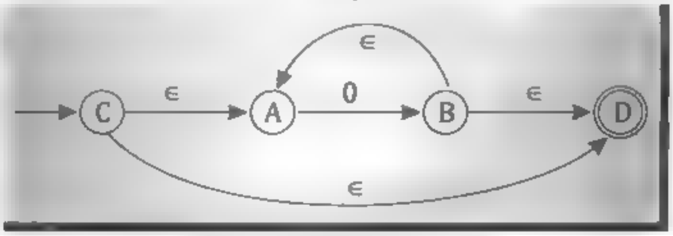
Construct on $\in$ – NFA for 0*

Solution

To accept 0



To accept 0*

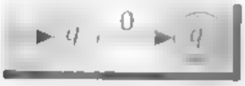


PROBLEM 4

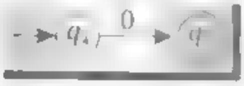
Construct on ∈ - NFA for the regular expression 00* + 1.

SOLUTION

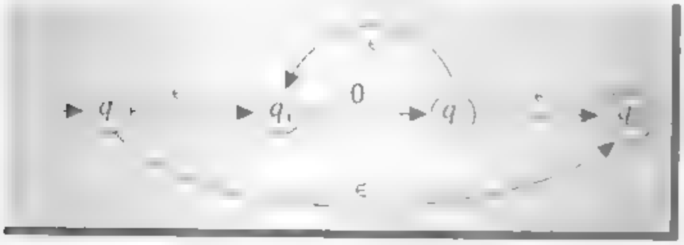
To accept 0



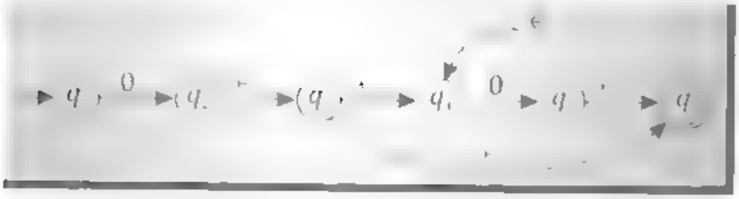
To accept 0



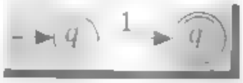
To accept 0*



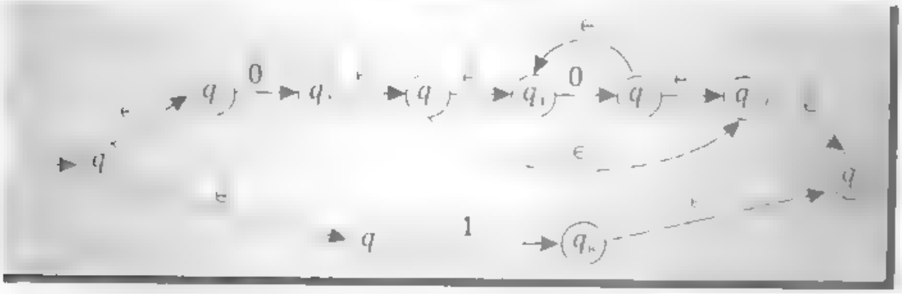
To accept 00*



To accept 1



To accept 00* + 1

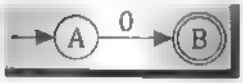


Problem 5

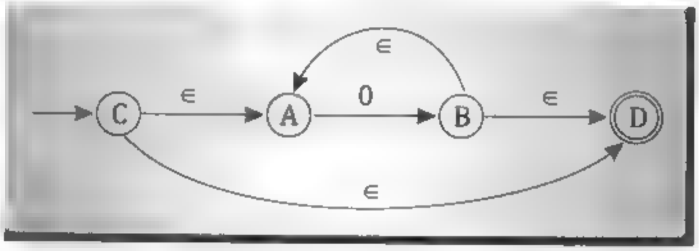
Construct a ∈ – NFA for (0* 0) + (1* 0)

Solution

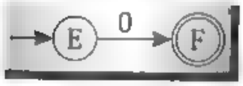
To accept 0



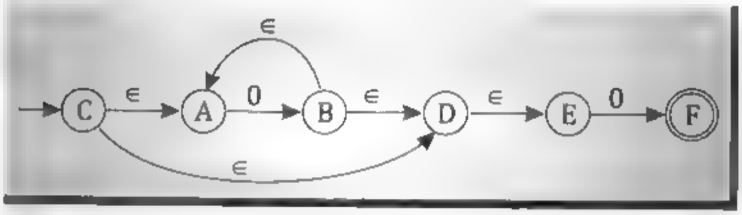
To accept 0*



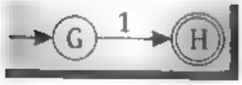
To accept 0



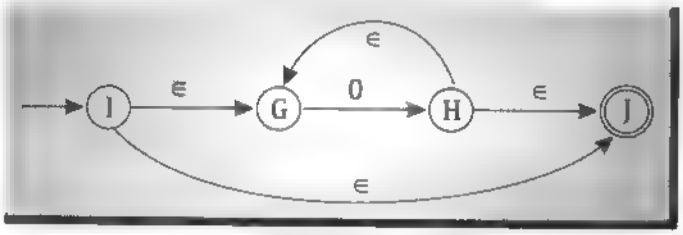
To accept 0*.0



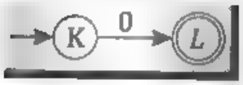
To accept 1



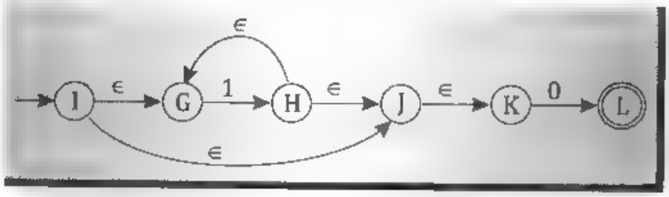
To accept 1*



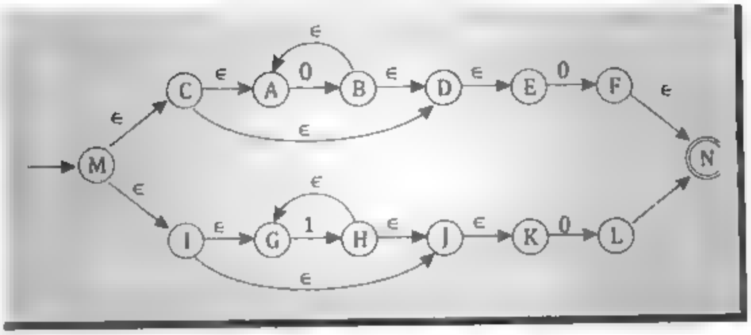
To accept 0



To accept 1*0



To accept (0*0) + (1*0)

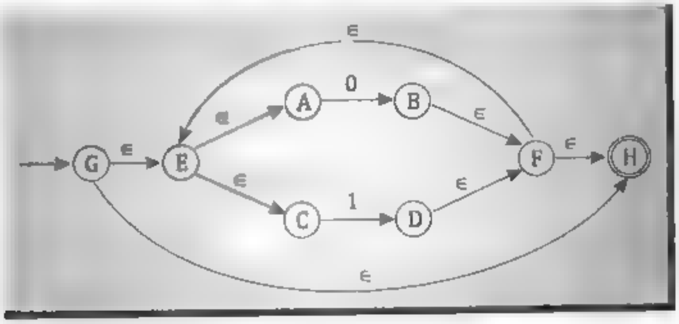


PROBLEM 6

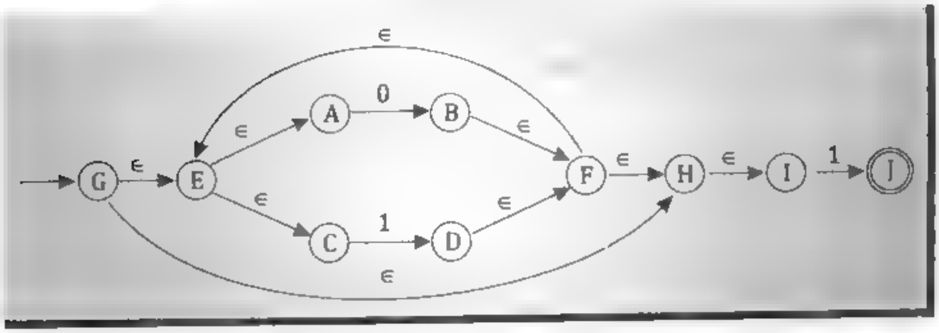
For the regular expression (0 + 1)* 1 (0 + 1) construct an ∈ – NFA.

SOLUTION

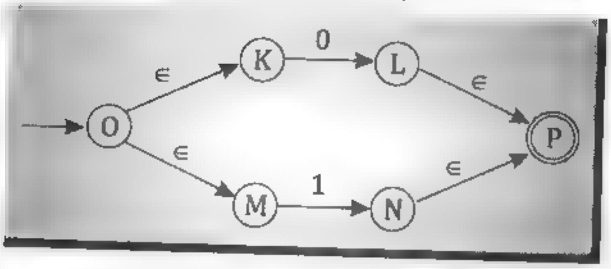
To accept (0 + 1)*



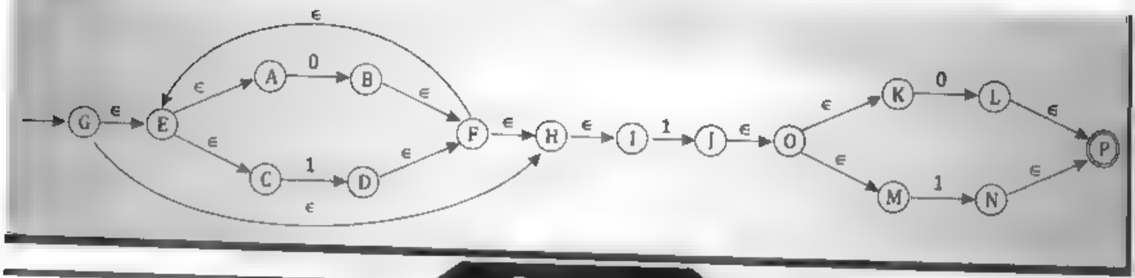
To accept (0 + 1)* 1



To accept (0 + 1)



To accept (0 + 1)* 1 (0 + 1)

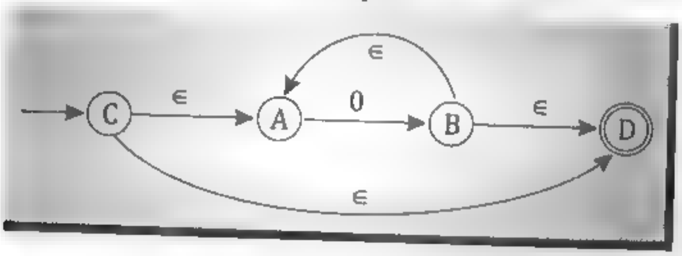


— PROBLEM 7 —

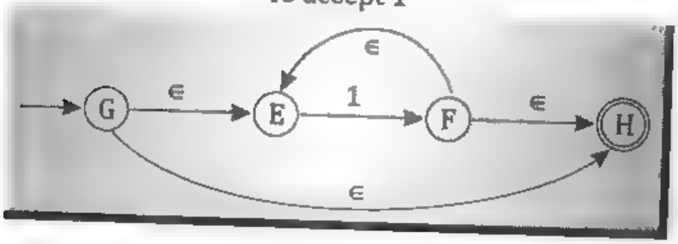
Construct on NFA with ∈ – moves for the regular expression (0* + 1*)* 2* 3*

SOLUTION

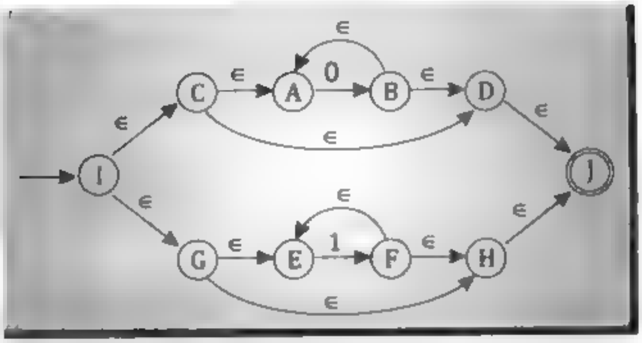
To accept 0*



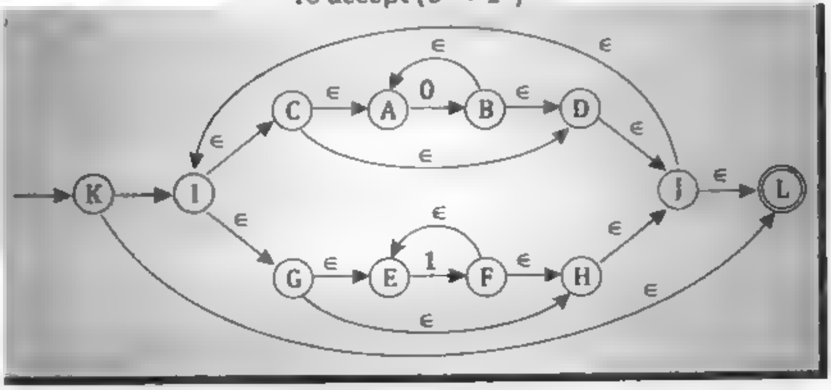
To accept 1*



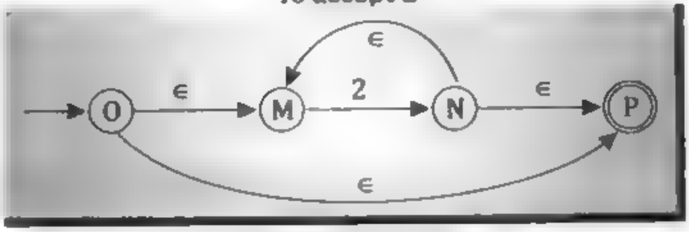
To accept 0* + 1*



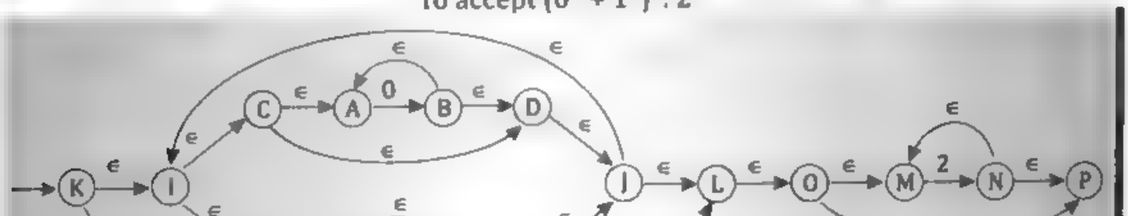
To accept (0* + 1*)*



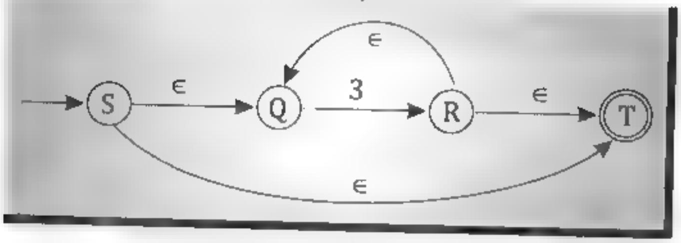
To accept 2*



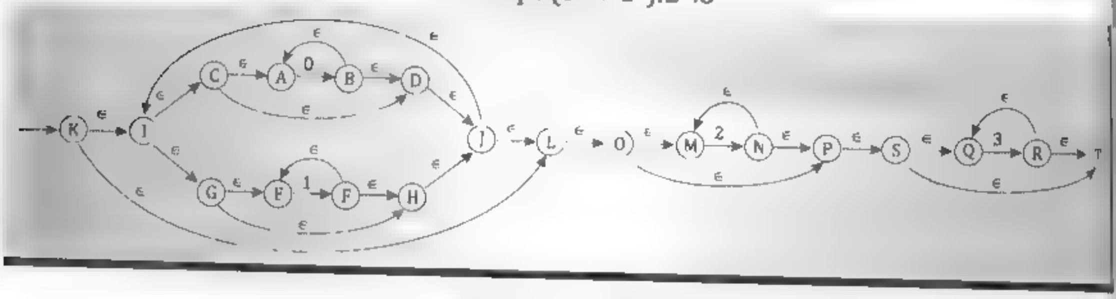
To accept (0* + 1*)*. 2*



To accept 3*



To accept (0* + 1*).2*.3*

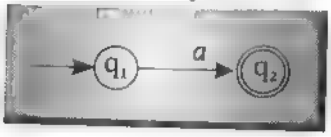


— Problem 8 —

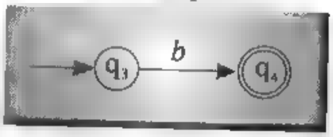
Construct an NFA with -moves for the RE (*a*+*b*)* *cd*

Solution

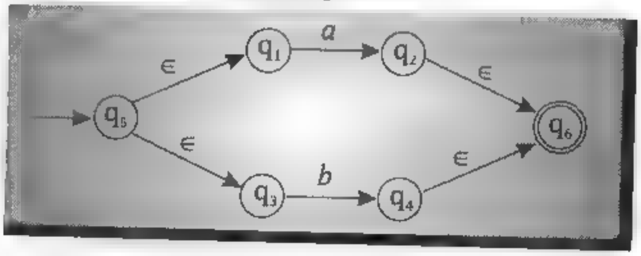
To accept *a*



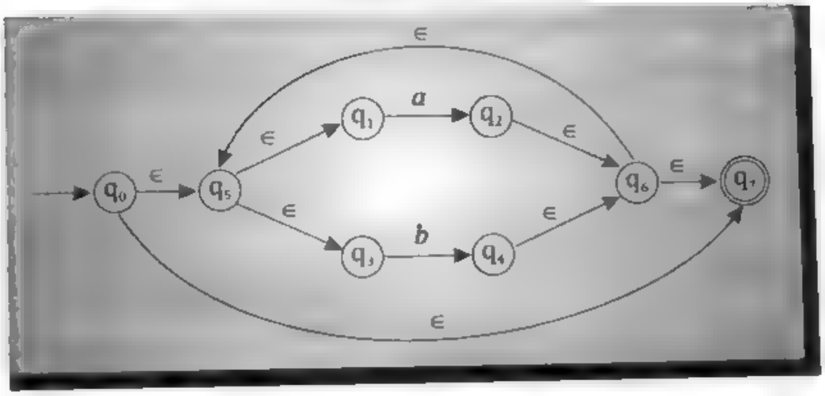
To accept *b*



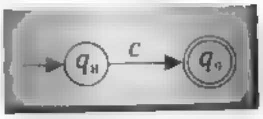
To accept *a* + *b*



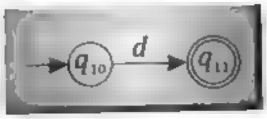
To accept $(a + b)^*$



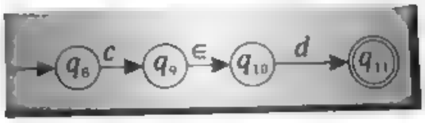
To accept c



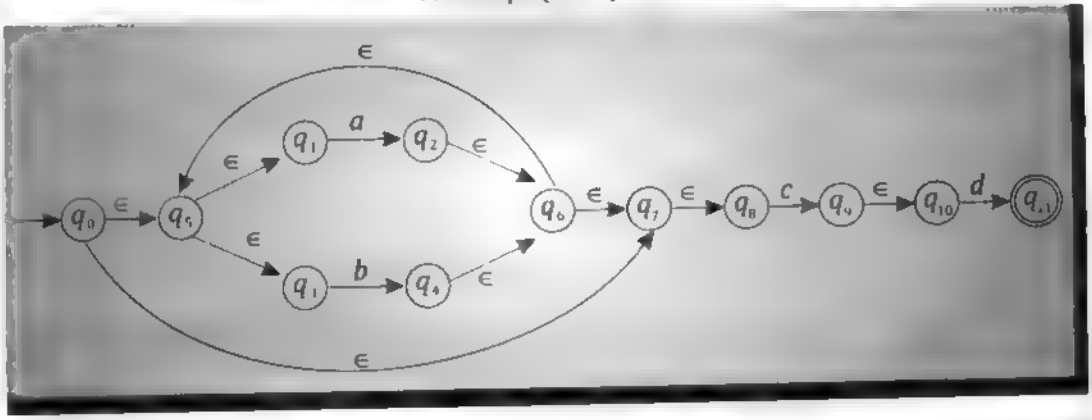
To accept d



To accept cd



To accept $(a + b)^* cd$



2.7 · Construction of Regular Expression from Finite Automa

Arden's Theorem: Let P and Q be two regular expressions over Σ . If P does not contain ∈, then the equation

$R = Q + RP$ has a unique solution. $R = QP^*$

Arden's theorem is used to find the regular expression from Finite Automata.

Algorithm to find regular expression from FA.

(i) Let q_1, be the initial state.

(ii) $q_2, q_3, \ldots\ldots q_n$ be the number of states.

(iii) q_j is the final state where $j \leq n$

(iv) Let α_{ji} represents transitions from q_j to q_i

(v) Calculate q_i such that $q_i = \alpha_{ji} . q_j$

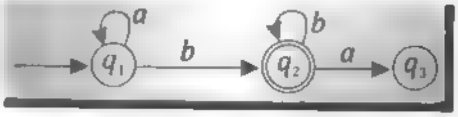
(vi) For the start state q_1, $q_1 = \alpha_{ji} \, q_j + \in$

(vii) Compute final state which ultimately gives the regular expression.

Note The above algorithm is applicable only for finite automata which does not have $\in$ - moves.

Problem 1

Construct Regular Expression from the given DFA



Solution

Step 1: q_1 **is the start state.**

Step 2: Calculate q_1 such that $q_1 = \alpha_{ji} \, q_j + \Sigma$ where α_{ji} is the transition from q_j to q_i.

$q_1 = q_1 \, a + \in$

$q_1 = \in \; q_1 \, a$

$q_1 = \in . \, a^*$ [$\because$ Arden Thm $R = Q + RP$ then $R = QP^*$]

$q_1 = a^*$ [$\because \in a^* = a^*$]

Step 3: Calculate q_i such that $q_i = \alpha_{ji} \, q_j$

$q_2 = q_1 \, b + q_2 \, b$

$q_2 = a^* \, b + q_2 \, b$

$q_2 = (a^* \, b) \, b^*$ [$\because$ R = Q + RP then R = QP*]

$q_2 = a^* \, bb^*$

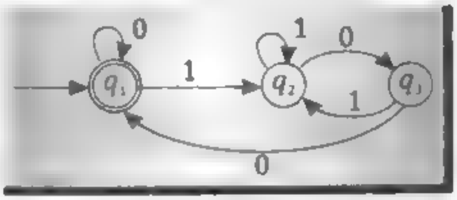
$q_2 = a^* \, b^+$ [$\because$ b $\cdot$ b* = b$^+$]

We don't want to find out q_3 equation because q_2 is the final state.

The regular expression is given by $a^* \, b^+$

Problem 2

Construct a regular expression from the given DFA



Solution

Step 1: q_1 is the start state

Step 2: Calculate q_1

$q_1 = q_1 0 + q_3 0 + \epsilon$

Step 3: Calculate q_2, q_3

$q_2 = q_2 \cdot 1 + q_3 \cdot 1 + q_1 \cdot 1$

$q_3 = q_2 \cdot 0$

Substitute q_3 in q_2

$q_2 = q_2 \cdot 1 + q_2 \cdot 10 + q_1 \cdot 1$

$q_2 = q_2 (1 + 01) + q_1 \cdot 1$

$q_2 = q_1 \cdot 1 + q_2 (1 + 01)$

$q_2 = q_1 \cdot 1 (1 + 01)^*$ $\quad$ [$\because$ If $R = Q + RP$ then $R = QP^*$]

Substitute in q_1

$q_1 = q_1 \cdot 0 + q_3 \cdot 0 + \epsilon$

$q_1 = q_1 \cdot 0 + q_2 \cdot 00 + \epsilon$

$q_1 = q_1 \cdot 0 + [q_1 \cdot 1 (1 + 01)^*]00 + \epsilon$

$q_1 = q_1 [0 + 1 (1 + 01)^* 00] + \epsilon$

$q_1 = \epsilon + q_1 [0 + 1 (1 + 01)^* 00]$

$q_1 = \epsilon [0 + 1 (1 + 01)^* 00]^*$ $\quad$ [By Arden's Theorem]

$q_1 = [0 + 1 (1 + 01)^* 00]^*$ $\quad$ [$\because \epsilon \cdot R = R$]

Since q_1 is the final state the required regular expression is $[0 + 1 (1 + 01)^* 00]^*$

$q_1 = q_0 0 + q_0 1 + q_2 0 + q_2 1$

$= \epsilon \cdot 0 + \epsilon \cdot 1 + q_2 (0 + 1)$ $\quad$ [$\because q_0 = \epsilon$]

$= 0 + 1 + q_2 (0 + 1)$ $\quad$ [$\because \epsilon \cdot R = R$]

$q_2 = q_1 0 + q_1 1$

$q_2 = q_1 (0 + 1)$

$= [\,0 + 1 + q_2\,(0 + 1)]\,(0 + 1)$

$q_2 = [(0 + 1)]\,(0 + 1)] + q_2\,[(0 + 1)]\,(0 + 1)]$

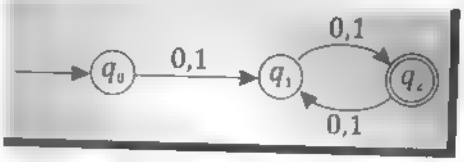
$q_2 = (0 + 1)\,(0 + 1)\,[(0 + 1)\,(0 + 1)]^*$ [$\because$ If $R = Q + RP$ then $R = QP^*$]

Since q_2 is the final state, the required regular expression is

$(0 + 1)\,(0 + 1)\,[\,(0 + 1)\,(0 + 1)]^*$

PROBLEM 3

Construct regular expression from the given DFA.



SOLUTION

Step 1: q_0 is the start state

Step 2: Calculate q_0

$q_0 = \epsilon$. Because q_0 does not get any insert.

Step 3: Calculate q_1 and q_2

$q_1 = q_0\,0 + q_0\,1 + q_1\,0 + q_1\,1$

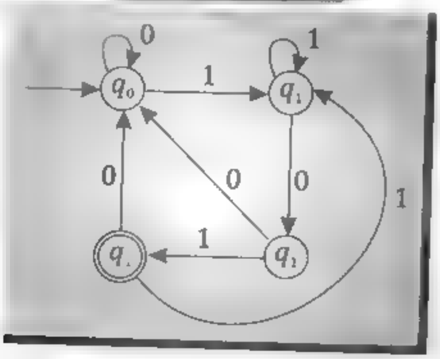
$q_2 = q_1\,0 + q_1\,1$

Since q_2 is the final state the required regular expression is $\mathbf{q_1 . 0 + q_1 . 1}$

PROBLEM 4

Construct regular expression for the given DFA.

SOLUTION



Step 1: q_0 is the start state

Step 2: Calculate q_0

$$q_0 = q_0 0 + q_2 0 + q_3 0 + q_1 + \in$$

Step 3: Calculate q_1, q_2, q_3

$$q_1 = q_0 1 + q_1 1 + q_3 1$$
$$q_2 = q_1 0$$
$$q_3 = q_2 \cdot 1$$

substitute q_2 in q_3

$$q_3 = q_1 01$$
$$q_1 = q_0 1 + q_1 1 + q_3 1$$

substitute q_3 in q_1

$$q_1 = q_0 1 + q_1 1 + q_1 011$$
$$q_1 = q_0 1 + q_1 (1 + 011)$$
$$q_1 = q_0 1 (1 + 011)^* \qquad [\because \text{If } R = Q + R P \text{ then } R = QP^*]$$
$$q_0 = q_0 0 + q_2 0 + q_3 0 + \in$$
$$= q_0 0 + (q_1 0) 0 + (q_1 01) 0 + \in \qquad [\because q_3 = q_1 01] \quad [\because q_2 = q_1 0]$$
$$= q_0 0 + q_1 [00 + (01) 0] + \in$$

substitute q_1 in the equation of q_0

$$q_0 = q_0 0 + [q_0 1 (1 + 011)^* (00 + 010)] + \in$$
$$q_0 = q_0 0 + q_0 [1 (1 + 011)^* (00 + 010)] + \in$$
$$q_0 = q_0 0 [0 + [1 (1 + 011)^* (00 + 010)] + \in$$
$$q_0 = \in + q_0 [0 + [1 (1 + 011)^* (00 + 010)] \qquad [\because \text{If } R = Q + R P \text{ then } R = QP^*]$$
$$q_0 = \in [0 + [1 (1 + 011)^* (00 + 010)]^*$$
$$q_0 = [0 + (1 + 011)^* (00 + 010)]^* \qquad [\text{ since } \Sigma \cdot R = R]$$

substitute q in q_3

$$q_3 = q_1 01$$
$$= q_0 1 (1 + 011)^* 01 [\because q_1 \text{ is substituted}]$$
$$q_3 = [0 + (1 + 011)^* (00 + 010)]^* 1 (1 + 011)^* 01$$

since q_3 is the final state the required regular expression is

$$\mathbf{[0 + (1 + 011)^* (00 + 010)]^* 1 (1 + 011)^* 01}$$

2.8 Application of Regular Expressions

Regular expressions are used in the following.

- Design of compilers
- To define languages

- Declarative way to express set of strings
- Validation – i.e., checking the correctness of inputs.
- Tokenisation – i.e., conversion of string of characters into a sequence of words later interpretation.
- In pattern matching
  - Test for a pattern within a string.
  - Replace text in a document
  - Extract a substring from a string based upon a pattern match
- Used in languages like Jscript and C for string handling
- Helps in implementing complex match logic in databases.

2.9 Pumping Lemma for Regular Languages

Pumping lemma is a method of pumping (generating) many input strings from a given string. It is used to show that certain languages are not regular.

2.9.1 – Theorem

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automation with n states. Let L be the regular language accepted by M. For every string x in L, there exists a constant n such that $|x| \geq n$.

Now, the string x can be broken into three substrings u, v and w such that, $x = u\,v\,w$.

Satisfying the following constraints

- $v \neq \in$
- $|uv| \leq n$

  then uv^iw is in L for $i \geq 0$

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automation and L be the regular language accepted by M.

Let $x = a_1, a_2, a_3 \text{-----} a_m$

Where $m \geq n$ and each a_i is in Σ. n is the number of states of the finite automata.

Since we have m input symbols, naturally we should have m+1 states in the sequence $q_0, q_1, q_2, \text{----} q_m$

Where q_0 will be the start state and q_m will be the final state as shown below.

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "a_1"] & q_1 \arrow[r, "a_2"] & q_2 \arrow[r, "a_3"] & {} & {} \arrow[r, "a_{m-1}"] & q_{m-1} \arrow[r, "a_m"] & \boxed{q_m}
\end{tikzcd}$$

Since $|x| \geq n$, by the pigeon hold principle it is not possible to have distinct transitions. One of the state can have a loop. Let the string x be divided into three substrings as shown below.

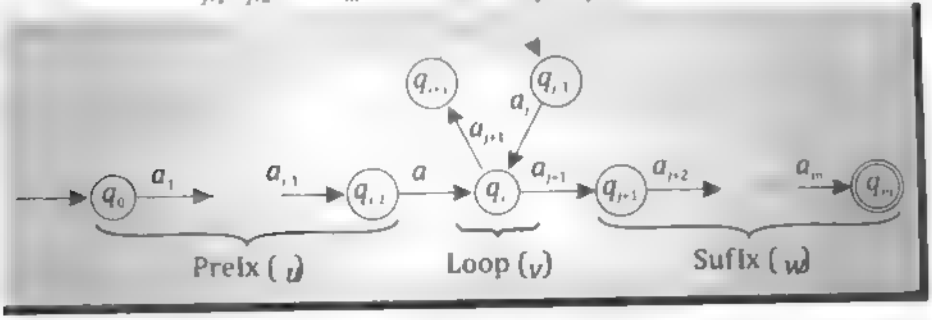
$x = u\,v\,w$

Where

$u = a_1 a_2 - - - - a_i$, prefix from $a_1 a_2 - - - - a_i$

$v = a_{i+1}, a_{i+2} - - - a_{j-1}, a_j$, loop string from $a_{i+1} a_{i+2} - - - a_{j-1} a_j$

$w = a_{j+1} a_{j+2} - - - a_m$, suffix from $a_{j+1}, a_{j+2} - - - a_m$



From the above figure, it is clear that the prefix string u takes the machine from q_0 to q_i, the loop string v takes the machine from q_i to q_i $[q_i = q_j]$ and suffix string w takes the machine from q_j to q_m.

The minimum string that can be accepted by the above FA is uw with $i = 0$.

But when $i = 1$, the string uvw is accepted by DFA.

When $i = 2$, the string $uvvw$ is accepted by DFA.

So, if $i > 0$, the machine goes from q_0 to q_i on input string u, circles from q_i to q_j based on the value of i and then goes to accepting state on input string w. In general, if the string x is split into substrings uvw, then for all $i \geq 0$,

$u\,v^i\,w \in L$

This can be expressed using the following transitions.

$$\begin{aligned}\delta(q_0, a_1 a_2 - - - a_{i-1} a_i a_{j+1} a_{j+2} - - - a_m) &= \delta(\delta(q_0, a_1, a_2 - - - a_{i-1} a_i), a_{j+1}, a_{j+2} - - - a_m)\\ &= \delta(q_i, a_{j+1} a_{j+2} - - - a_m)\\ &= \delta(q_k, a_{k+1} a_{k+2} - - - a_m)\\ &= q_m\end{aligned}$$

Also, after the string $a_1, a_2 - - - - a_i$ the machine will be in state q_i. Since q_i and q_j are same, we can input the string $a_{i+1}\ a_{i+2} - - - - a_j$ any number of times and the machine will stay in q only. Finally if the input string is $a_{j+1}\ a_{j+2} - - - a_m$ the machine enters into final state q_m.

2.10 Application of Pumping Lemma

Pumping Lemma can be sued to prove that certain languages are not regular.

Steps to prove that certain languages are not regular

Step 1: Assume that the language L is regular. Let n be the number of states in th corresponding FA.

Step 2: Choose a string 'x' such that $|x| \geq n$. Use pumping lemma to write $x = uvw$ wit the constraints

- $V \neq \in$., $|v| \geq 1$
- $|uv| \leq n$

Step 3: Find a suitable integer i such that $u\,v^i w \notin L$. According to pumping lemma, $u\,v$ is in L for $i \geq 0$. So, the result is contradiction to the assumption that the language regular.

Therefore, the given language L is not regular.

Problem 1

Show that $L = \{w\,w^R | w \in (a+b)^*\}$ is not regular.

Solution

Step 1. Suppose L is regular. Let n be the number of states in the FA accepting L.

Consider the string

$$x = \overbrace{a \quad a\,b \quad b}^{w}\;\overbrace{b \quad b\,a \quad a}^{w^R}$$

Where $w = a\_\_\_\,a\,b\_\_\_\,b$ and reverse of w, $w^R = b\_\_\_\,b\,a\_\_\_a$

Step 2: Since $|x| \geq n$, we can split the string x into uvw such that $/uv/ \leq n$ and $/v/ \geq 1$

$$x = \underbrace{a\_\_\_}_{u}\,\underbrace{a}_{v}\,\underbrace{b\_\_\_\,b\;b\_\_\_b\,a\_\_\_\,a}_{w}$$

Where $\lvert u\rvert$	$= n\text{-}1$ and $\lvert v\rvert = 1$
So that $\lvert uv\rvert$	$= \lvert u\rvert + \lvert v\rvert$
	$= n - 1 + 1$
	$= n$, which is true.

According to pumping lemma, $u\,v^i w \in L$ for $i = 0, 1, 2\_\_\_\_$

Step 3: If i is 0, that is v does not appear and so the number of a's on the left of x wil be less than the number of b's on the right of x and so the string is not of the form ww^R

So $u v^i w \notin L$, when $i = 0$

This is a contradiction to the assumption that the language is regular. So, the language

$L = \{ww^R \mid w \in (a + b)^*\}$ is not regular.

Problem 2

Show that the language [L_{eq}] consisting the strings with an equal number of 0's followed by equal number 1's is not a regular language.

OR

Prove that, $L_{eq} = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Solution

$L_{eq} = \{ 01, 0011, 000111, \ldots \}$

Step 1: Let L_{eq} be regular and n be the number of states in FA. Consider the string, $x = 0^n 1^n$.

Step 2: Since $|x| = 2n \geq n$

we can split x into uvw such that $|uv| \leq n$ and $|v| \geq 1$

$$x = \underbrace{000}_{u}\ \underbrace{0}_{v}\ \underbrace{1111}_{w}$$

where $|u| = n - 1$ and $|v| = 1$

so that $|uv| = |u| + |v|$

$= n - 1 + 1$

$= n$

and $|w| = n$

According to pumping lemma, $u v^i w \notin L$ for $i = 0, 1, 2, \ldots$.

Step 3: If i is 0, the string v does not appear and so the number of 0's will be less than the number of 1's and so that string x does not contain n number of 0's followed by n number of 0's followed by n number of 1's.

But, according to pumping lemma, n number of 0's should be followed by n number of 1's, which is a contradiction to the assumption that the language is regular.

So, the language

$L_{eq} = \{0^n 1^n \mid n \geq 0\}$ is not regular.

PROBLEM 3

Show that the language, $L = \{a^{k^2} | K \text{ is an integer}, k \geq 1\}$

Which consists of all strings of a's whose length is a perfect square, is not regular.

SOLUTION

Step 1: Let L be regular and n be the number of states in FA. Let us choose the value of which depends on n.

That is, Let $x = a^m \in L$ Where $m = n^2$

Step 2: Note that $|x| \geq n$ and so, we can split x into uvw such that $|uv| \leq n$ and $|v| \geq 1$ a shown below

$$x = a^m$$

$$= \underbrace{a^j}_{u}\underbrace{a^k}_{v}\underbrace{a^{m-j-k}}_{w}$$

Where $|u| = j$ and $|v| = k \geq 1$ and so $|uv| = |u| + |v| = j + k \leq n$

Step 3 : According to pumping lemma, $u\, v^i w \in L$, for $i = 0, 1, 2, \_\_\_$

i.e., $a^j a^{ki} a^{m-j-k} \in L$ for $i = 0, 1, 2 \_\_\_$

Now, if we choose $i = 2$, we have

$$u\, v^2 w \in L$$

$$a^j a^{2k} a^{m-j-k} \in L$$

$$a^{m+k} \in L$$

since $k \geq 1$, we have

$$|a^{m+k}| = m + k$$

$$= n^2 + k \text{ (since } m = n^2)$$

Note that

$$n^2 < n^2 + k < n^2 + 1 \text{ (when } k = 1)$$

$$< (n^2 + 2n + 1) = (n + 1)^2$$

$$\text{so } n^2 < (n + 1)^2$$

Since $n^2 + k$ (nothing but $m + k$) lies between n^2 and $(n + 1)^2$, it is not a perfect squar which is contradiction to the assumption that it should be a perfect square.

So the language

$$L = \{a^{k^2} \mid k \text{ is an integer}, k \geq 1\} \text{ is not regular.}$$

Problem 4

Show that $L = \{w \mid n_a(w) = n_b(w)\}$ is not regular

Solution

The language consists of the strings with an equal number of a's and b's

i.e., $L = \{ \in, ab, ba, abab, aabb, bbaa, \_\_\_\_\_ \}$

Step 1: Let L be regular and n be the number of states in FA. Let us choose the value of x which depends on n.

Let $x = a^n b^n \in L$

Step 2: Note that $|x| = 2n \geq n$ and so, we can split x into uvw such that $|uv| \leq$ n and $|v| \geq 1$ as shown below

$x = a^m$

Where $|u| = j$ and $|v| = k \geq 1$ and so that $|uv| = |u| + |v| = j + k \leq n$

Step 3 : According to pumping lemma, $u v^i w \in L$ for $i = 0, 1, 2, \_\_\_\_$

ie., $a^j (a^k)^i b^n \in L$ for $i = 0, 1, 2, \_\_\_\_$

Now, if we choose $i = 0$, the number of a's will be less than the number of b's and if we choose $i = 2$, uv^2w will have more a's than b's which is contradiction to the assumption that it has equal number of a's and b's.

So the language

$L = \{ w|n_a(w) = n_b(w) \}$ is not regular.

Problem 5

Show that $L = \{ w|n_a(w) < n_b(w) \}$ is not regular.

Solution

The language consists of strings a's and b's, where the number of a's is less than the number of b's.

Step 1: Let L be regular and n be the number of states in FA. Let us choose the value of x which depends on n.

Let $x = a^{n-1} b^n \in L$

Step 2: Note that $|x| = 2n - 1 \geq$ n

$\therefore$ we can split x into uvw.

Such that $|uv| \leq n$ and $|v| \geq 1$ as

$$x = a^{n-1}b^n$$
$$= \underbrace{a^{n-1}}_{u}\underbrace{b^k}_{v}\underbrace{b^{n-k}}_{w}$$

where $|u| = n - 1$ and $|v| = K$

$\therefore \quad |uv| = |u| + |v| = n\text{-}1 + K \le n$

In this case the value of K should be less than or equal to 1.

Step 3: According to pumping lemma, $u\,v^i\,w \in L$ for $i = 0, 1, 2, \_\_\_\_$

ie., $a^{n-1}\,(b^k)^i\,b^{n-k} \in L$ for $i = 0, 1, 2, \_\_\_$

Now, if we choose $i = 0$,

$a^{n-1}\,b^{n-k} \in L$

and when $k = 1$, $a^{n-1}\,b^{n-k} \in L$

So when $i = 0$, the number of a's and b's are same, which is a contradiction t the assumption that the number of a's will be less than the number of b's. So, th language $L = \{w|n_a(w) < n_b\,(w)\}$ is not regular.

PROBLEM 6

Show that $L = \{\,a^i\,b^j \mid i > j\,\}$ is not regular

SOLUTION

The language consists of strings of a's and b's, where the number of a's are more than th number of b's.

Step 1: Let L be regular and n be the number of states in FA. Consider the string

$$x = a^{n+1}\,b^n$$

Step 2: Since $|\,x\,| = 2n{+}1 \ge$ n, we can split x into uvw such that $|uv| \le n$ and $|v| \ge 1$

$$x = a^{n+1}b^n$$
$$= \underbrace{a^j}_{u}\,\underbrace{a^k}_{v}\,\underbrace{ab^n}_{w}$$

where $|u| = j$ and $|v| = k \ge 1$

$\therefore \quad |uv| = |u| + |v| = j + k \le n$

Step 3: According to pumping lemma,

$u\,v^i\,w \in L$ for $i \ge 0$

ie., $a^j\,(a^k)^i\,a\,b^n \in L$ for $i \ge 0$

Now, if we choose $i = 0$, number of a's in string u will not be more than number of b's in w, which is a contradiction to the assumption that number of a's are more than the number of b's.

So, the language

$$L = \{\,a^i\,b^j \mid i > j\,\} \text{ is not regular}$$

Problem 7

Show that $L = \{ a^{n!} / n \geq 0 \}$ is not regular

Solution

$n! = 1*2*3* \_\_\_ n$

Step 1 Let L be regular and n be the number of states in FA.

Step 2: Let $x = a^{n!}$

It is clear that $|x| \geq n$.

So we can split x into uvw, such that $|uv| \leq n$ and $/v/ \geq 1$ as shown below

$x = a^{n}$

$= \underbrace{a^{j}}_{u} \underbrace{a^{k}}_{v} \underbrace{a^{n \quad j \quad k}}_{w}$

where $|u| = j$ and $|v| = k \geq 1$

$|uv| = |u| + |v| = j + k \leq n$

Step 3 According to pumping lemma, $u v^{i} w \in L$, for $i = 0, 1, 2, \_\_\_\_$

Ie., $a^{j} (a^{k})^{i} a^{n!-j+k} \in L$

if $i = 0$

$a^{j} a^{n!-j+k} \in L$

$a^{n!-k} \in L$

It is very clear that $n! > n! - k$

when $k = 1$, $n! > n! - 1$

But according to pumping lemma, $n! = n! - 1$ which is not and is a contradiction. So L can not be regular. So, the language,

$L = \{a^{n!} \mid n \geq 0\}$ is not regular.

Problem 8

Show that $L = \{0^{n}/n$ is prime$\}$ is not regular

Solution

Prime numbers are 1, 3, 5, 7, ____ so the language, $L = \{0, 000, 00000, \_\_\_\_\_\}$

Step 1: Let L be regular and n be the number of states in FA.

Let $x = 0^{n} \in L$, where n is prime.

Step 2: $|x| = n$

Therefore we can split x into uvw such that $/uv/ \leq n$ and $/v/ \geq 1$ as shown below.

$$x = 0^n$$
$$= \underbrace{0^j}_{u}\underbrace{0^k}_{v}\underbrace{0^{n-j-k}}_{w}$$

where $|u| = j$ and $|v| = k \geq 1$

$|uv| = |u| + |v| = j + k \leq n$

Step 3 : According to pumping lemma, $u\,v^i\,w \in L$ for $i = 0, 1, 2$ ____

i.e., $0^j\,(0^k)^i\,0^{n-j-k} \in L$

i.e., $j + ki + n - j - k = n + k\,(i - 1)$ is prime for all $i \geq 0$

If we choose $i = n + 1$ then

$$n + k\,(i - 1) = n + kn$$
$$= n\,(k + 1)$$

is also a prime for each $k \geq 1$ which is a contradiction, because if $k = 1$ it will not b a prime.

So, the language $L = \{0^n \mid n \text{ is prime}\}$ is not regular.

Problem 9

Show that $L = \{ww|w \in \{a, b\}^*\}$ is not regular.

Solution

Step 1. Let L be regular and n be the number of states in FA. Let r us choose the value of which depends on n.

i.e., $x = a^n\,b\,a^n\,b \in L$

Step 2: Note that $|x| = 2n + 2 \geq n$ and so we can split x into uvw such that $|uv| \leq n$ and $|v| \geq$ as shown below

$$x = a^n b a^n b$$
$$= \underbrace{a^j}_{u}\underbrace{a^k}_{v}\underbrace{ba^n b}_{w}$$

where $|u| = j$ and $|v| = k \geq 1$ and so that

$|uv| = |u| + |v|$

$= j + k \leq n$

Step 3. According to pumping lemma, $u\,v^i\,w \in L$ for $i = 0, 1, 2$ ____

ie., $a^j\,(a^k)^i\,b\,a^n\,b \in L$ for $i = 0, 1, 2,$ ____

Now if we choose $i = 0$, number of a's on the left of first b will be less than the number of a's after the first b which is a contradiction to the assumption that the are not equal.

So the language $L = \{\,ww|w \in \{a, b\}^*\}$ is not regular.

2.11 PROPERTIES OF REGULAR LANGUAGES

The regular languages exhibit two types of properties. They are

1. Closure properties. Ex.: Union, Intersection, etc.,
2. Decision properties. Ex.: Whether 2 Finite Automata accept same languages or not.

2.11.1 Closure properties

Some new languages can be constructed from already existing languages using certain operations such as union, intersection, concatenation etc. These properties are called closure properties. They are

- Union of two Regular languages is regular
- Intersection of two Regular languages is regular
- Complement of a Regular language is regular
- Closure of a Regular language is regular
- Concatenation of two Regular language is regular
- Difference of two Regular languages is regular
- Reversal of Regular language is regular
- A homomorphism of a Regular language is regular
- The inverse homomorphism of a Regular language is regular

(a) Regular Languages are closed under Union, Intersection and Star (closure)

Theorem 2.3 :

Show that if L_1 and L_2 are regular then $L_1 \cup L_2$, $L_1 \cdot L_2$ and L_1^* are also regular languages.

Proof: It is given that L_1 and L_2 are regular languages. So, there exists regular expressions R_1 and R_2 such that $L_1 = L(R_1)$ $L_2 = L(R_2)$

By definiton of regular expressions we have

- $R_1 + R_2$ is a regular expression denoting the Langauge $L_1 \cup L_2$.
- $R_1 \cdot R_2$ is a regular expression denoting the language $L_1 \cdot L_2$
- R_1^* is a regular expression denoting the language L_1^*

So regular languages are closed under union, concatenation and closure operations.

(b) Closure under complementation:

Theorem 2.4

Show that if L is a Regular language then complement of L is also regular.

Proof: Let $M_1 = (Q, \Sigma, \delta_1, q_0, F)$ be a FA which accepts language L.

$M_2 = (Q, \Sigma, \delta_2, q_0, Q - F)$ be a FA which accepts language L which is a complement of language L.

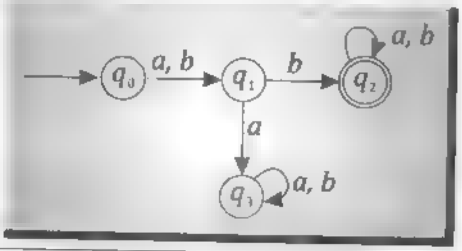
Since the final state of M_1 is the non-final state of M_2 and non-final state of M_1 is the final state of M_2. There exists M_2 which accepts $\overline{L}$. So $\overline{L}$ is also Regular language.

Example :

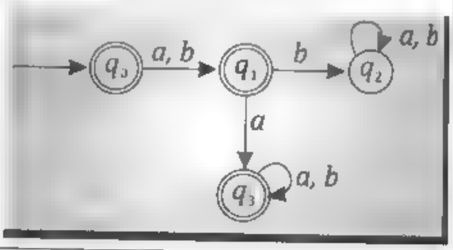
L = { All words with length of atleast two and second letter as b } and $\Sigma = \{ a, b\}$

$\overline{L}$ = { All words with length less than two or second letter not b }

Let M_1 be the FA which accepts L



Then M_2 is the FA which accepts $\overline{L}$



(c) Closure under Intersection.

Therem 2.5: Show that if L_1 and L_2 are Regular then it is closed under Intersection

Proof: L_1 and L_2 are regular languages. Let $L = L_1 \cap L_2$

Applying De morgan's theorem

$L_1 \cap L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ $[\because \overline{\overline{\overline{L_1} \cap \overline{L_2}}} = \overline{\overline{L_1} \cup \overline{L_2}}]$

Since L_1, L_2 is regular

⇒ $\overline{L_1}$, $\overline{L_2}$ is regular [$\because$ regular sets are closed under complementation]

⇒ $\overline{L_1} \cup \overline{L_2}$ is regular [$\because$ regular sets are closed under union]

⇒ $\overline{\overline{L_1} \cup \overline{L_2}}$ is regular [$\because$ regular sets are closed under complement]

⇒ $L_1 \cap L_2$ is regular (By demorgan's law)

So regular languages are closed under intersection.

(d) Closure under difference

Theorem 2.6

If L_1 and L_2 are regular languages then the regular languages is closed under difference. (i.e.,) $L_1 - L_2$ is regular.

Proof:

Since L_1, L_2 is regular language, then $\overline{L_2}$ is also regular.

⇒ $L \cap M$ is also regular. [$\because$ regular languages are closed under intersection]

⇒ $L - M$ is also regular [$\because L - M = L \cap \overline{M}$]

(e) **Closure under reverse**

Theorem 2.7

If L is regular L^R is also regular.

Proof

Let L be the regular language corresponding to regular expression E.

It is required to prove that there is another regular expression E^R such that

$$L(E^R) = (L(E))^R$$

Basis

By definition of regular expression E,

- ϕ is a regular expression
- $\in$ is a regular expression
- a is a regular expression

So the reversal of regular expression E^R is given by

- $\{\phi\}^R = \{\phi\}$ • $\{\in\}^R = \{\in\}$ • $\{a\}^R = \{a\}$

Induction

By definition of regular expressions, if E_1 and E_2 are regular expressions then,

- $E_1 + E_2$ is a regular expression
- $E_1 \cdot E_2$ is a regular expression
- $E_1{}^*$ is a regular expression

which results in three different cases. Let us prove each of these.

Case 1: If $E = E_1 + E_2$ is a regular expression then $E^R = E_1^R + E_2^R$ is a regular expression denoting the language $L\left(E^R\right) = L\left(E_1^R\right) \cup L\left(E_2^R\right)$

Case 2: If $E = E_1 \cdot E_2$ is a regular expression, then $E^R = E_1^R + E_2^R$ is a regular expression denoting the languages $L\left(E^R\right) = L\left(E_1^R\right) \quad L\left(E_2^R\right)$

Case 3: If $E = E_1^*$ is as regular expression then $E^R = (E_1^*)^R$ is a regular expression denoting the languages $L(E^R) = L\left((E_1^*)^R\right)$

Closure under Homomorphism

Definition

Let Σ and Γ be the set of alphabets. The homomorphic function

$$h : \Sigma \to \Gamma^*$$

is called homomorphism (i.e.,) a substitution where a single letter is replaced by a string. If

$$w = a_1\, a_2\, a_3 \_\_\_\_ a_n$$

$$h(w) = h(a_1)\, h(a_2)\, h(a_3) \_\_\_\_\_ h(a_n)$$

If L is made of alphabets from Σ, then $h(L) = \{\, h(w) \mid w \in L \,\}$ is called homomorphic image.

Theorem 2.8

If L is regular and h is homomorphism, then homomorphic image $h(L)$ is regular.

Proof:

Let R be regular expression and $L(R)$ be the corresponding regular language. We can find $h(R)$ by substituting $h(a)$ for each a in Σ. By definition of regular expression, $h(R)$ is regular expression and $h(L)$ is regular language.

So regular language is closed under homomorphism.

2.11.2 Decision Properties

A Decision property for a class of language is an algorithm that takes a formal description of a language and tells whether or not some property hold.

Example :

(i) **Whether the language is empty**

(ii) **Whether the language is finite**

(iii) Whether the given string belongs to the language

(iv) Whether the language a subset of another regular language

(v) Whether the language is same as another regular language

Let us proof some of the decision properties.

(i) Whether the language is empty (i.e.,) $L = \phi$.

Use DFA representation. Use a graph reachability algorithm to test if atleast one accepting state is reachable from the start state.

(ii) **Whether the language is finite.**

Given a DFA, for the language eliminate all the states that are not reachable from the start state and all the states that do not reach accepting states. Test if any cycle exists, if so L is infinite otherwise L is finite.

(iii) Do two finite automata accept the same language (or) Is the language same as **another regular language.**

This can be proved with minimal finite automata.

Given FA create a new FA with a minimal number of states that accept the same language. Generate minimal automata for each FA. Then compare on a state by state basis.

2.12 Minimization of Finite Automata

For storage efficiency, it is desirable to represent the DFA with fewer states, hence it is required to minimize the DFA. The minimization of finite automa is very important in the design of switching circuit, as the number of states of the automa implemented by the circuit decreases, the cost of the circuit also decreases.

Minimization of automa can be achieved by finding distinguishable and indistinguishable states.

Equivalence of two states: Two states P and q are equivalent (indistinguishable) if and only if $\delta(P, w)$ and $\delta(q, w)$ are final states or both $\delta(P, w)$ and $\delta(q, w)$ are non final states for $w \in \Sigma^*$ (i.e.,) if $\delta(P, w) \in F$ and $\delta(q, w) \in F$ or $\delta(P, w) \notin F$ and $\delta(q, w) \notin F$ then the states P and q are indistinguishable.

If there is atleast one string w such that one of $\delta(P, w)$ and $\delta(q, w)$ is final state and the other is non-final, then the states the states P and q are distinguishable (unquivalent) states.

Table Filling Algorithm: Table filling algorithm is used to find set of states that are distinguishable and indistinguishable.

The algorithm is recursively defined as follows.

Step 1: For each pair (p, q) where $p \in Q$ and $q \in Q$, if $P \in F$ and $q \notin F$ or vice versa, then the pair (p, q) is distinguishable and mark the pair by putting 'x'

Step 2: Identify the subsequent markings

For each pair (p, q) and for each $a \in F$ find $\delta(p, a) = r$ and $\delta(p, a) = S$. If the pair (r, s) is already marked as distinguishable and mark it as 'x'

Repeat step 2 until no previously unmarked pairs are marked.

Step 3 : If the pair (p,q) obtained using the above table filling algorithm are indistinguishable, then the two state P and q are equivalent and they can be merged into one state.

Minimization of DFA: The algorithm to minimize the DFA is as follows.

Step 1: Find distinguishable and indistinguishable pairs using the table filling algorithm.

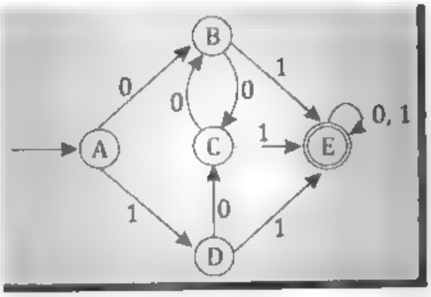
Step 2: Obtain states of minimuzed DFA wihch includes both indistinguishable pairs and distinguishable state.

Step 3: Compute the transition table.

If $[P_1, P_2, \_\_\_\_ P_k]$ is a group and if $\delta([P_1, P_2, \_\_\_\_ P_k], a) = [r_1, r_2 \_\_\_\_ r_m]$ then place an edge from $[P_1, P_2, \_\_\_ P_k]$ to $[r_1, r_2, \_\_\_\_ r_m]$ and label the edge with the symbol a. Follow this procedure for all the states obtained in step 2 for each $a \in \Sigma$.

Problem 1

Reduce the DFA



Solution

The transition table of the above DFA

δ	0	1
→A	B	D
B	C	E
C	B	E
D	C	E
* E	E	E

The DFA can be minimized using table filling algorithm.
The various states of the DFA are A, B, C, D, E.

Step 1:

(a) Draw the table

(i) Vertically write all the states from second state to last state like B, C, D

(ii) Horizontally write all the states from first state to last but one state like
B, C, D

	B				
	C				
	D				
*	E				
		A	B	C	D

(b) Initial Marking

Since E is the final state, the pairs (A, E), (B, E), (C, E), (D, E) has one final state and other one non-final state. So, the pairs are marked horizontally.

	B				
	C				
	D				
*	E	×	×	×	×
		A	B	C	D

Step 2: Subsequent marking

For each pair (p, q) and for each $a \in \Sigma$ find $\delta(p, a) = r$ and $\delta(p, a) = s$. If pair (r, s) is already marked as "distinguishable" then pair (p, q) is also distinguishable state and mark it as 'x'.

δ	0	1	
(A, B)	(B, C)	(D, E)	(D, E) is marked, so mark (A, B)
(A, C)	(B, B)	(D, E)	(D, E) is marked, so mark (A, C)
(A, D)	(B, C)	(D, E)	(D, E) is marked, so mark (A, D)
(B, C)	(C, B)	(E, E)	
(B, D)	(C, C)	(E, E)	
(C, D)	(B, C)	(E, E)	

B	×			
C	×			
D	×			
* *E*	×	×	×	×
	A	*B*	*C*	*D*

δ	0	1
(*B*, *C*)	(*C*, *B*)	(*E*, *E*)
(*B*, *D*)	(*C*, *C*)	(*E*, *E*)
(*C*, *D*)	(*B*, *C*)	(*E*, *E*)

Repeat Step 2

Since there is no further marking, stop the iteration.

Indistinguishable pairs (*B*, *C*) (*B*, *D*), (*C*, *D*) = (*B*, *C*, *D*)

Distinguishable states = *A*, *E*

Step 3: Obtain the states of minimized DFA

A, {*B*, *C*, *D*}, *E*

Step 4: **Identify the start state**

Since *A* is the start state of DFA, A is the start state of minimized DFA.

Step 5: **Identify the final state**

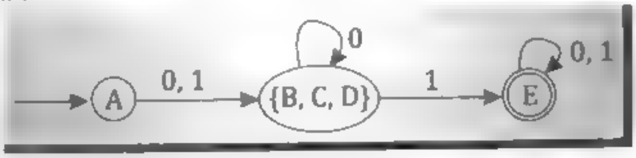
Since *E* is the final state of DFA, E is the final state of minimized DFA.

Step 6: Compute the transition table

δ	0	1
→ *A*	{*B*, *C*, *D*}	{*B*, *C*, *D*}
{*B*, *C*, *D*}	{*B*, *C*, *D*}	*E*
* *E*	*E*	*E*

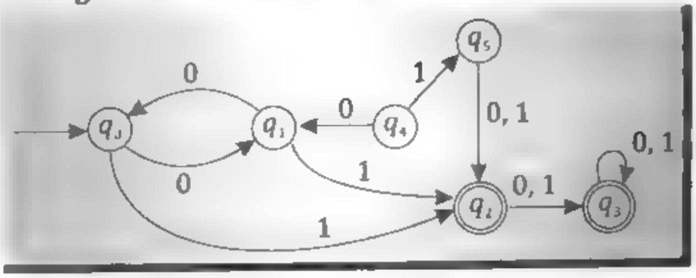
Minimized DFA is shown below

the equivalent



— Problem 2 —

Minimize the following DFA



SOLUTION

The states q_4 and q_5 are not reachable from start state. So they have to be removed. The various states of the DFA are q_0, q_1, q_2, q_3. The transition table of the DFA is

Step 1: Draw the table

	0	1
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_2
$* \; q_2$	q_3	q_3
$* \; q_3$	q_3	q_3

Step (ii) Initial marking

	q_1			
*	q_2			
*	q_3			
		q_0	q_1	q_2
				*

Since q_2 and q_3 are final states, mark the pairs (q_0, q_2), (q_1, q_2), (q_0, q_3), (q_1, q_3). Do not mark (q_2, q_3) Since both are final states.

	q_1			
*	q_2	×	×	
*	q_3	×	×	
		q_0	q_1	q_2
				*

Step 2: Subsequent marking

δ	0	1
(q_0, q_1)	(q_1, q_0)	(q_2, q_2)

There is no further marking, so we can stop the iteration.

Indistinguishable pairs = (q_0, q_1) (q_2, q_3)

Distinguishable state = Null

Step 3: Obtain the states of minimised DFA $\{q_0, q_1\}$, $\{q_2, q_3\}$

Step 4: Identify the start state

Since q_0 is the start state of DFA, the pair (q_0, q_1) is the start state of minimized DFA.

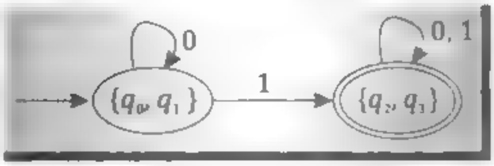
Step 5: Identify the final state

Since q_2 and q_3 are the final states of DFA, the pair (q_2, q_3) is the final state of Minimised DFA.

Step 6: Compute transition table

	0	1
$\rightarrow \{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_2, q_3\}$
$^*\{q_2, q_3\}$	$\{q_2, q_3\}$	$\{q_2, q_3\}$

Minimized DFA is shown below



Problem 3

Find the minimized DFA from the following transition table

δ	a	b
$\rightarrow A$	B	A
B	A	C
C	D	B
$^* D$	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Solution

The various states of the DFA re A, B, C, D, E, F, G, H.

Step 1: Draw the table and do initial marking.

Since D is the final state, the pairs (D, E), (D, F), (D, G), (D, H) are marked vertically and (A, D), (B, D), (C, D) are marked horizontally.

B							
C							
* D	x	x	x				
E				x			
F				x			
G				x			
H				x			
	A	B	C	D *	E	F	G

Step 2: Subsequent marking

δ	a	b	
(A, B)	(B, A)	(A, C)	
(A, C)	(B, D)	(A, B)	(B, D) is marked, so mark (A, C)
(A, E)	(B, D)	(A, F)	(B, D) is marked, so mark (A, E)
(A, F)	(B, G)	(A, E)	
(A, G)	(B, F)	(A, G)	
(A, H)	(B, G)	(A, D)	(A, D) is marked, so mark (A, H)
(B, C)	(A, D)	(C, B)	(A, D) is marked, so mark (B, C)
(B, E)	(A, D)	(C, F)	(A, D) is marked, so mark (B, E)
(B, F)	(A, G)	(C, E)	
(B, G)	(A, F)	(C, G)	
(B, H)	(A, G)	(C, D)	(C, D) is marked, so mark (B, H)
(C, E)	(D, D)	(B, F)	
(C, F)	(D, G)	(B, E)	(D, G) is marked, so mark (C, F)
(C, G)	(D, F)	(B, G)	(D, F) is marked, so mark (C, G)
(C, H)	(D, G)	(B, D)	(D, G) is marked, so mark (C, H)
(E, F)	(D, G)	(F, E)	(D, G) is marked, so mark (E, F)
(E, G)	(D, F)	(F, G)	(D, F) is marked, so mark (E, G)
(E, H)	(D, G)	(F, D)	(D, G) is marked, so mark (E, H)
(F, G)	(G, F)	(E, G)	
(F, H)	(G, G)	(E, D)	(E, D) is marked, so mark (F, H)
(G, H)	(F, G)	(G, D)	(G, D) is marked, so mark (G, H)

Update the new marking into the table.

B							
C	x	x					
D	x	x	x				
E	x	x		x			
F			x	x	x		
G			x	x	x		
H	x	x	x	x	x	x	x
	A	B	C	D	E	F	G

Repeat step 2 and find subsequent marking.

δ	*a*	*b*	
(*A*, *B*)	(*B*, *A*)	(*A*, *C*)	(*A*, *C*) is marked so mark (*A*, *B*)
(*A*, *F*)	(*B*, *G*)	(*A*, *E*)	(*A*, *E*) is marked, so mark (*A*, *F*)
(*A*, *G*)	(*B*, *F*)	(*A*, *G*)	
(*B*, *F*)	(*A*, *G*)	(*C*, *E*)	
(*B*, *G*)	(*A*, *F*)	(*C*, *G*)	(*C*, *G*) is marked, so mark (*B*, *G*)
(*C*, *E*)	(*D*, *D*)	(*B*, *F*)	
(*F*, *G*)	(*G*, *F*)	(*E*, *G*)	(*E*, *G*) is marked, so mark (*F*, *G*)

Update the new marking into the table.

B	x						
C	x	x					
* *D*	x	x	x				
E	x	x		x			
F	x		x	x	x		
G		x	x	x	x	x	
H	x	x	x	x	x	x	x
	A	*B*	*C*	*D*	*E*	*F*	*G*
				*			

Repeat step 2 and identify further marking.

δ	*a*	*b*
(*A*, *G*)	(*B*, *F*)	(*A*, *G*)
(*B*, *F*)	(*A*, *G*)	(*C*, *E*)
(*C*, *E*)	(*D*, *D*)	(*B*, *F*)

Since there is no further marking, we can stop the iteration.

Indistinguishable pairs = (*A*, *G*), (*B*, *F*), (*C*, *E*)

Distinguishable states = *D*, *H*

Step 3: Obtain the states of minimised DFA

{*A*, *G* }, {*B*, *F*}, {*C*, *E* }, *D*, *H*

Step 4: Identify the start state.

Since A is the start state, the pair (*A*, *G*) is the start state of minimized DFA.

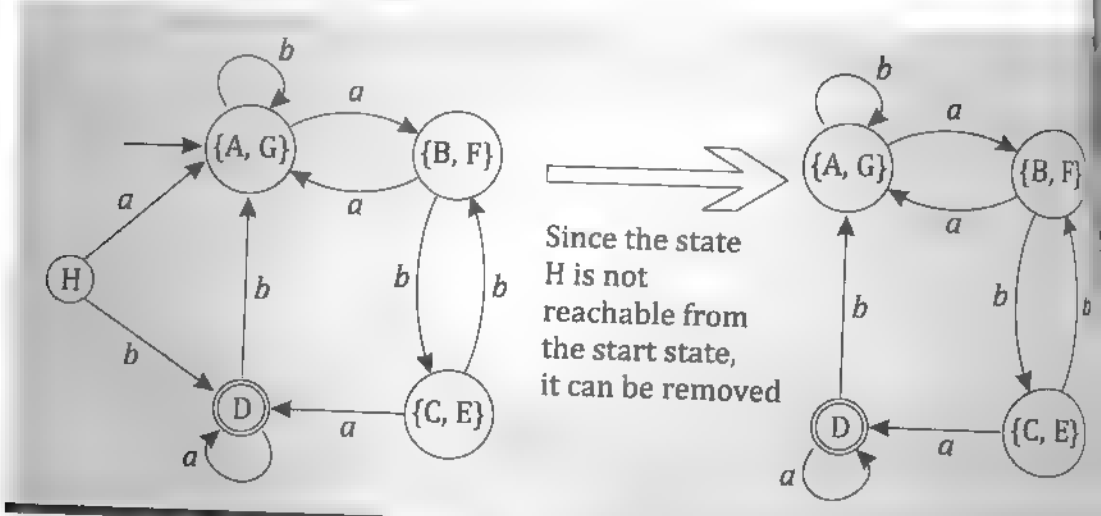
Step 5: Identify final state.

F = { *D* }

Step 6: Compute the transition table.

δ	*a*	*b*
→ {*A*, *G*}	{*B*, *F*}	{*A*, *G*}
{*B*, *F*}	{*A*, *G*}	{*C*, *E*}
{*C*, *E*}	*D*	{*B*, *F*}
* *D*	*D*	{*A*, *G*}
H	{*A*, *G*}	*D*

Minimized DFA is shown below



Since the state H is not reachable from the start state, it can be removed

— Problem 4 —

Minimize the following DFA

	δ	*a*	*b*
→	*A*	*B*	*E*
	B	*C*	*F*
*	*C*	*D*	*H*
	D	*E*	*H*
	E	*F*	*I*
*	*F*	*G*	*B*
	G	*H*	*B*
	H	*I*	*C*
*	*I*	*A*	*E*

SOLUTION

The various states of DA are *A, B, C, D, E, F, G, H, I*

Step 1: Draw the table and Do initial markings

- For the state C, the pairs (*D, C*), (*E, C*), (*G, C*) and (*H, C*) are marked vertically and (*A, C*) and (*B, C*) are marked horizontally.
- For the state F, the pairs (*G, F*) and (*H, F*) are marked vertically and (*A, F*), (*B, F*)
- (*D, F*), (*E, F*) are marked horizontally
- For the state I, the pairs (*A, I*), (*B, I*), (*D, I*), (*E, I*) (*G, I*) and (*H, I*) are marked horizontally.

  Do not mark the pairs (*C, I*), (*C, F*), (*F, I*) as *C, F, I* all are final states.

	B	x							
*	*C*	x	x						
	D			x					
	E			x					
*	*F*	x	x		x	x			
	G			x			x		
	H			x			x		
*	*I*	x	x		x	x		x	x
		A	*B*	*C*	*D*	*E*	*F*	*G*	*H*
				*			*		

Step 2: Subsequent marking

δ	*a*	*b*	
(*A, B*)	(*B, C*)	(*B, F*)	(*B, C*) is marked so mark (*A, B*)
(*A, D*)	(*B, E*)	(*B, H*)	
(*A, E*)	(*B, F*)	(*B, I*)	(*B, F*) is marked so mark (*A, E*)
(*A, G*)	(*B, H*)	(*B, B*)	
(*A, H*)	(*B, I*)	(*B, C*)	(*B, C*) is marked, so mark (*A, H*)
(*B, D*)	(*C, E*)	(*F, H*)	(*C, E*) is marked, so mark (*B, D*)
(*B, E*)	(*C, F*)	(*F, I*)	
(*B, G*)	(*C, H*)	(*F, B*)	(*C, H*) is marked, so mark (*B, G*)
(*B, H*)	(*C, I*)	(*F, C*)	
(*D, E*)	(*E, F*)	(*H, I*)	(*H, I*) is marked, so mark (*D, E*)
(*D, G*)	(*E, H*)	(*H, B*)	
(*D, H*)	(*E, I*)	(*H, C*)	(*E, I*) is marked, so mark (*D, H*)
(*E, G*)	(*F, H*)	(*I, B*)	(*F, H*) is marked, so mark (*E, G*)
(*E, H*)	(*F, I*)	(*I, C*)	
(*H, G*)	(*I, H*)	(*C, B*)	(*C, B*) is marked, so mark (*H, G*)

Update the new marking into the table

	A	B	C	D	E	F	G	H
B	x							
* C	x	x						
D		x	x					
E	x		x	x				
* F	x	x		x	x			
G		x	x		x	x		
H	x		x	x		x	x	
* I	x	x		x	x		x	x

repeat step 2 to identify further marking

δ	a	b
(A, D)	(B, E)	(B, H)
(A, G)	(B, H)	(B, B)
(B, E)	(C, F)	(F, I)
(B, H)	(C, I)	(F, C)
(D, G)	(E, H)	(H, B)
(E, H)	(F, I)	(I, C)

Since, there is no further marking, we can stop the iteration.

Indistinguishable pairs are

(A, D), (A, G), (D, G) = (A, D, G)

(B, E), (B, H), (E, H) = (B, E, H)

(C, F), (C, I), (F, I) = (C, F, I)

distinguishable states does not exists.

Step 2: The start states of minimized DFA are { A, D, G }, { B, E, H }, { C, F, I }

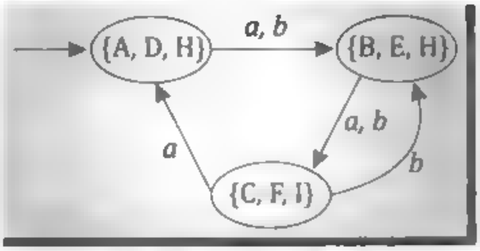
Step 3: The state of minimized DFA = { A, D, G }

Step 4: The final state of minimised DFA = { C, F, I }

Step 5: Compute the transition table

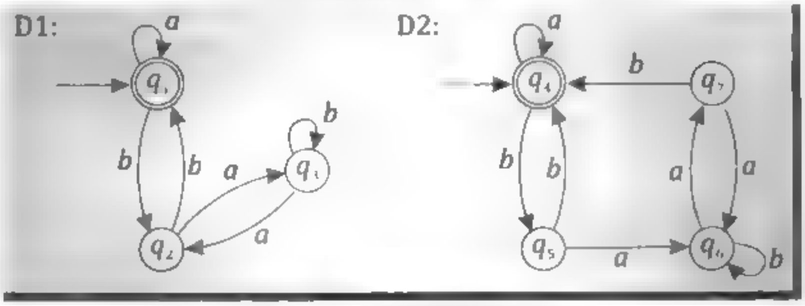
δ	a	b
→ { A, D, H }	{ B, E, H }	{ B, E, H }
{ B, E, H }	{ C, F, I }	{ C, F, I }
* { C, F, I }	{ A, D, H }	{ B, E, H }

Minimized DFA is shown below



Problem 5

Check whether the following two DFA's are equivalent.



Solution

First we will reduce the DFA D1.

δ	a	b
→* q_1	q_1	q_2
q_2	q_3	q_1
q_3	q_2	q_1

The various states of DFA D1 are q_1, q_2, q_3.

Step 1: Draw the table and do the initial marking.

Since q_1 is the final state, and q_2, q_3 are non final states, the pairs (q_1, q_2), (q_1, q_3) are marked vertically.

q_2	x	
q_3	x	
	q_1	q_2

*

Step 2: Subsequent marking

δ	a	b
(q_2, q_3)	(q_1, q_2)	(q_1, q_3)

(q_1, q_3) is marked so mark (q_2, q_3)

Update the new marking into the table.

q_2	x	
q_3	x	x
	q_1	q_2

Since there is no further marking, we can stop the iteration.

From the table, we can notice that there is no Indistinguishable pairs. So the DFA cannot be reduced.

Reduce the second DFA D2.

δ	a	b
$\rightarrow^* q_4$	q_4	q_5
q_5	q_6	q_4
q_6	q_7	q_6
q_7	q_6	q_4

The various states of DFA D2 are q_4, q_5, q_6, q_7.

Step 1: Draw the table and do the initial marking.

Since q_4 is the final state and q_5, q_6, q_7 are non final states, the pairs $(q_4, q_5), (q_4, q_6), (q_4, q_7)$ are marked vertically.

q_5	x		
q_6	x		
q_7	x		
	q_4	q_5	q_6

Step 2: Subsequent marking

	a	b
(q_5, q_6)	(q_6, q_7)	(q_4, q_6)
(q_5, q_7)	(q_6, q_6)	(q_4, q_4)
(q_6, q_7)	(q_7, q_6)	(q_6, q_4)

(q_4, q_6) is marked so mark (q_5, q_6)

(q_6, q_4) is marked so mark (q_6, q_7)

Update the new marking into the table

q_5	x		
q_6	x	x	
q_7	x		x
	q_4	q_5	q_6

*

Repeat step 2 to identify further marking

	a	b
(q_5, q_7)	(q_6, q_6)	(q_4, q_4)

Since there is no further marking, we can stop the iteration.

Indistinguishable pairs are (q_5, q_7)

Distinguishable states are q_4, q_6

Step 3: The states of minimised DFA D_2^1 are q_4, $\{q_5, q_7\}$, q_6.

Step 4: The start state of minimized DFA $D_2^1 = \{q_4\}$

Step 5: The final state of minimized DFA $D_2^1 = \{q_4\}$

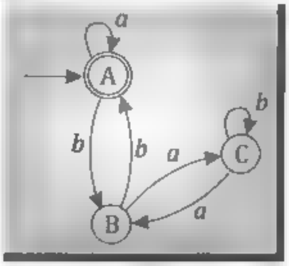
Step 6: Compute the transition table of minimized DFA D_2^1.

δ	a	b
$\rightarrow * q_4$	q_4	q_5
$\{q_5, q_7\}$	q_6	q_4
q_6	$\{q_5, q_7\}$	q_6

Renaming the states q_4 as A, $\{q_5, q_7\}$ as B q_6 as C, the transition table becomes

δ	a	b
$\rightarrow * A$	A	B
B	C	A
C	B	C

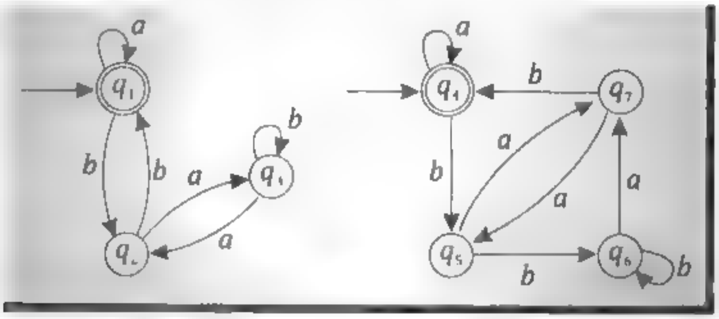
Minimized DFA D_2^1 is shown below



Since all the transitions and states of DFA D_1 and D_2^1 are similar, we can say that the given DFA D_1 and D_2 are equivalent.

Problem 6

Check whether the following DFA's are equivalent.



Solution

From the previous problem we know DFA D_1 cannot be minimized further. Let us reduce DFA D2.

δ	a	b
$\to * q_4$	q_4	q_5
q_5	q_7	q_6
q_6	q_7	q_6
q_7	q_5	q_4

The various states of DFA D_2 are q_4, q_5, q_6, q_7.

Step 1: Draw the table and do initial marking.

Since q_4 is the final state and q_5, q_6, q_7 are non final states the pairs (q_4, q_5), (q_4, q_6), (q_4, q_7) are marked vertically.

q_5	x		
q_6	x		
q_7	x		
	q_4 *	q_5	q_6

Step 2: Subsequent marking

	a	b	
(q_5, q_6)	(q_7, q_7)	(q_6, q_6)	
(q_5, q_7)	(q_7, q_5)	(q_6, q_4)	**(q_6, q_4) is marked so mark (q_5, q_7)**
(q_6, q_7)	(q_7, q_5)	(q_6, q_4)	**(q_6, q_4) is marked so mark (q_6, q_7)**

Update the new marking into the table.

q_5	x		
q_6	x		
q_7	x	x	x
	q_4	q_5	q_6

*

Repeat step 2 to identify further marking.

	a	b
(q_5, q_6)	(q_7, q_7)	(q_6, q_6)

Since there is no further marking, we can stop the iteration.

Indistinguishable pairs are (q_5, q_6)

Distinguishable states are q_4, q_7

Step 3: The states of minimized DFA D_2^1 are q_4, (q_5, q_6), q_7

Step 4: The start states of minimized DFA $D_2^1 = \{q_4\}$

Step 5: The start states of minimized DFA $D_2^1 = \{q_4\}$

Step 6: Compute the transition table.

δ	a	b
$\rightarrow^* q_4$	q_4	$\{q_5, q_6\}$
$\{q_5, q_6\}$	q_7	$\{q_5, q_6\}$
q_7	$\{q_5, q_6\}$	q_4

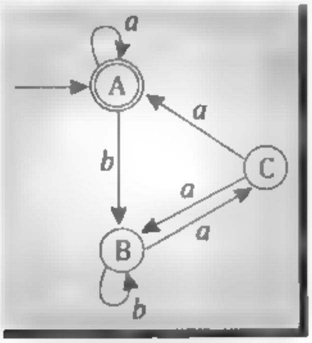
Renaming the states

q_4	as	A
$\{q_5, q_6\}$	as	B
q_7	as	C

The transition table becomes

δ	a	b
$\rightarrow^* A$	A	B
B	C	B
C	B	A

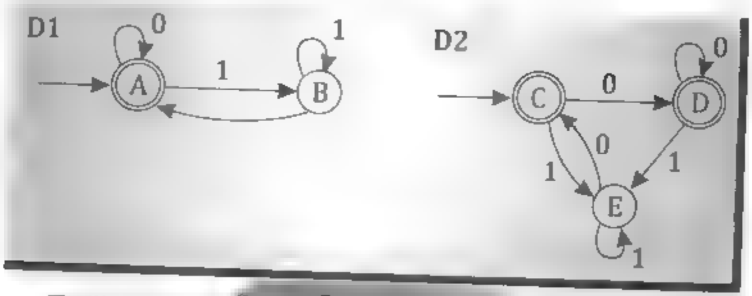
The minimized DFA D_2^1 is shown below



Since the transitions of DFA D_1 and D_2^1 are different, we can conclude the given DFA D_1 and D_2^1 are not equivalent.

Problem 7

Are the following automata equivalent



Solution

For the DFA D_1, A is the final state and B is non-final state, which are not equal. So DFA D can not be minimized.

Let us reduce DFA D_2.

δ	0	1
$\rightarrow * C$	D	E
$*D$	D	E
E	C	E

The various states of DFA D_2 are C, D and E

Step 1: Draw the table and do initial marking.

Since C and D are final states and E is non final state mark the pairs (C, E) and (D, E). Do not mark (C, D) as both C and D are final states.

*	D		
	E	x	x
		C	D
		*	*

Step 2: Since both C and D are final states.

States there is no. subsequent marking and we can stop the iteration.

Indistinguishable pairs are (C, D) Distinguishable State E.

Step 3: The states of minimized DFA D_2^1 are (C, D), E

Step 4: The start state of minimized DFA $D_2^1 = \{C, D\}$

Step 5: The final state of minimized DFA $D_2^1 = \{C, D\}$

Step 6: Compute the transition table

δ	0	1
$\rightarrow * \{C, D\}$	$\{C, D\}$	E
E	$\{C, D\}$	E

Renaming the states

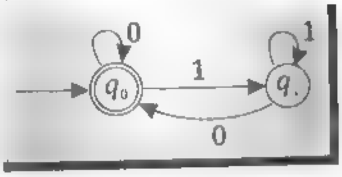
(C, D) as q_0

E as q_1

The transition table becomes

δ	0	1
$\rightarrow *q_0$	q_0	q_1
q_1	q_0	q_1

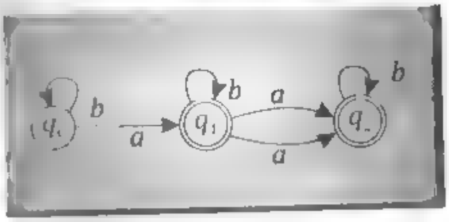
Minimized DFA D_2^1 is shown below



Since all the transitions and states of DFA D1 and D_2^1 are similar, we can conclude that the given DFA D_1 and D_2 are equivalent.

PROBLEM 8

Minimize the following DFA



SOLUTION

The transition table of the above DFA

δ	a	b
$\rightarrow q_0$	q_1	q_0
$* q_1$	q_2	q_1
$* q_2$	q_1	q_2

The various states of the DFA are q_0, q_1, q_2.

Step 1: **(i)** Draw the table

*	q_1		
*	q_2		
		q_0	q_1
			*

(ii) Initial Marking

Since q_1 and q_2 are the final states mark the pairs (q_0, q_2) since both are final state

*	q_1	x	
*	q_2	x	
		q_0	q_1
			*

Step 2: Subsequent marking.

Both q_1 and q_2 are final states of there is no further marking, so we can stop thi iteration.

Indistinguistable pair = (q_1, q_2)

Distinguishable state = q_0

Step 3: Obtain the states of minimized DFA q_0, $\{q_1, q_2\}$

Step 4: Identify the start state q_0 is the start state of minimized DFA.

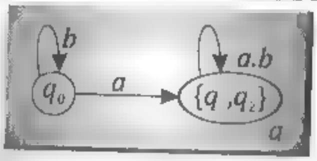
Step 5: Identify the final state .

Since q_1 and q_2 are the final states of DFA, the pair (q_1, q_2) is the final state of minimized DFA.

Step 6: Compute the transition table

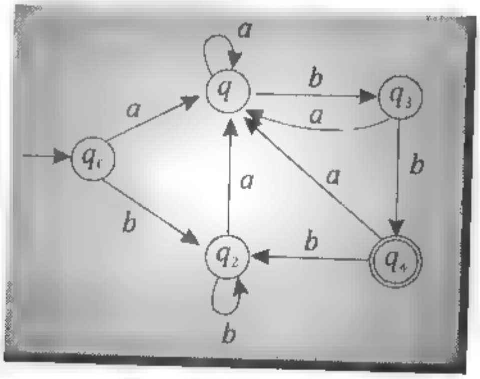
δ	a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	q_0
$*\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$

Minimized DFA is shown below



— Problem 9 —

Minimize the following DFA



SOLUTION

The transition table of the above DFA

δ	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
$* q_4$	q_1	q_2

The various states of the DFA are q_0, q_1, q_2, q_3, q_4

Step 1: Draw the table with initial marking

q_1				
q_2				
q_3				
$* q_4$	x	x	x	x
	q_0	q_1	q_2	q_3

Step 2 : Subsequent marking

δ	a	b	
(q_0, q_1)	(q_1, q_1)	(q_2, q_3)	
(q_0, q_2)	(q_1, q_1)	(q_2, q_2)	
(q_0, q_3)	(q_1, q_1)	(q_2, q_4)	(q_2, q_4) is marked. So mark (q_0, q_3)
(q_1, q_2)	(q_1, q_1)	(q_3, q_2)	
(q_1, q_3)	(q_1, q_1)	(q_2, q_4)	(q_2, q_4) is marked. So mark (q_1, q_3)
(q_2, q_3)	(q_1, q_1)	(q_2, q_4)	(q_2, q_4) is marked. So mark (q_2, q_3)

q_1				
q_2				
q_3	x	x	x	
$* q_4$	x	x	x	x
	q_0	q_1	q_2	q_3

Repeat Step 2 :

δ	a	b	
(q_0, q_1)	(q_1, q_1)	(q_2, q_3)	(q_2, q_3) is marked. So mark (q_0, q_1)
(q_0, q_2)	(q_1, q_1)	(q_2, q_2)	
(q_1, q_2)	(q_1, q_1)	(q_3, q_2)	(q_3, q_2) is marked. So mark (q_1, q_2)

q_1	x			
q_2		x		
q_3	x	x	x	
* q_4	x	x	x	x
	q_0	q_1	q_2	q_3

Repeat Step 2 :

δ	a	b
(q_0, q_2)	(q_1, q_1)	(q_2, q_2)

Since there is no further marking, stop the interation.

Indistinguishable pairs : (q_0, q_2)

Distinguishable states : q_1, q_3, q_4

Step 3 : Obtain the states of minimized DFZ

$\{q_0, q_2\}$ q_1, q_3, q_4

Step 4 : Identify the start states

Since q_0 is the start state of DFA, the pair $\{q_0, q_2\}$ is the start state of minimized DFA

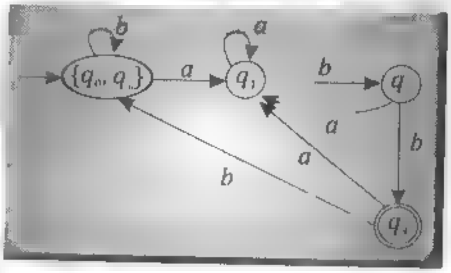
Step 5 : Identify the final state.

Since q_4 is the final state of DFA, q_4 is the final state of minimized DFA.

Step 6 : Compute transition table

δ	a	b
$\rightarrow \{q_0, q_2\}$	q_1	$\{q_0, q_2\}$
q_1	q_1	q_3
q_3	q_1	q_4
* q_4	q_1	$\{q_0, q_2\}$

Minimized DFA is shown below



Minimize the following DFA

δ	0	1
$\rightarrow q_0$	q_1	q_2
q_1	q_2	q_3
q_2	q_2	q_4
* q_3	q_3	q_3
* q_4	q_4	q_4
q_5	q_5	q_5

SOLUTION

The various states of the DFA are $q_0, q_1, q_2, q_3, q_4, q_5$

Step 1 : Draw the table with initial marking

	q_1					
	q_2					
*	q_3	x	x	x		
*	q_4	x	x	x		
	q_5				x	x
		q_0	q_1	q_2	q_3	q_4
					*	*

Step 2: Subsequent marking

δ	a	b	
(q_0, q_1)	(q_1, q_2)	(q_2, q_3)	(q_2, q_3) is marked. So mark (q_0, q_1)
(q_0, q_2)	(q_1, q_2)	(q_2, q_4)	(q_2, q_4) is marked. So mark (q_0, q_2)
(q_0, q_5)	(q_1, q_5)	(q_2, q_4)	(q_2, q_4) is marked. So mark (q_0, q_5)
(q_1, q_2)	(q_2, q_2)	(q_3, q_4)	
(q_1, q_5)	(q_2, q_5)	(q_3, q_4)	
(q_2, q_5)	(q_2, q_5)	(q_4, q_4)	

	q_1	x				
	q_2	x				
*	q_3	x	x	x		
*	q_4	x	x	x		
	q_5	x			x	x
		q_0	q_1	q_2	q_3	q_4
					*	*

Repeat Step 2 :

δ	a	b
(q_1, q_2)	(q_2, q_2)	(q_3, q_4)
(q_1, q_5)	(q_2, q_5)	(q_3, q_4)
(q_2, q_5)	(q_2, q_5)	(q_4, q_4)

Since there is no further marking, we stop the iteration.

Indistinguishable pairs : (q_1, q_2), $(q_2,$ (q_1, q_5), (q_3, q_4)

$= (q_1, q_2, q_5), (q_3, q_4)$

Distinguishable state : q_0

Step 3: The state of minimized DFA

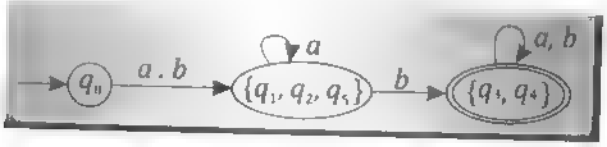
$q_0, \{q_1, q_2, q_5\}, \{q_3, q_4\}$

Step 4: q_0 is the start state of minimized DFA

Step 5: Since q_3 and q_4 are the final states of DFA, the pairs $\{q_3, q_4\}$ is the final state minimized DFA

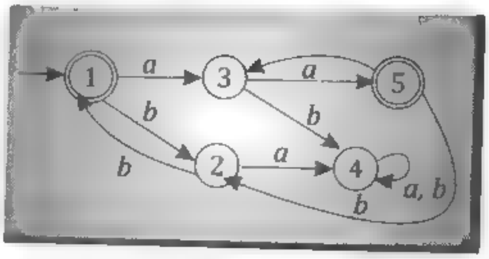
δ	a	b
$\rightarrow q_0$	(q_1, q_2, q_5)	(q_1, q_2, q_5)
(q_1, q_2, q_5)	(q_1, q_2, q_5)	(q_3, q_4)
$*(q_3, q_4)$	(q_3, q_4)	(q_3, q_4)

Step 6:



PROBLEM 11

Minimize the following DFA



SOLUTION

Transition table of the above DFA

δ	*a*	*b*
→* 1	3	2
2	4	1
3	5	4
4	4	4
* 5	3	2

The various states of the DFA are 1, 2, 3, 4, 5

Step 1: Draw the table with initial marking

	2	*x*			
	3	*x*			
	4	*x*			
*	5		*x*	*x*	*x*
		1	2	3	4
		*			

Step 2: Subsequent marking

δ	a	b
(2, 3)	(4, 5)	(1, 4)
(2, 4)	(4, 4)	(1, 4)
(3, 4)	(5, 4)	(4, 4)

(1, 4) is marked. So mark (2, 3)
(1, 4) is marked. So mark (2, 4)
(5, 4) is marked. So mark (3, 4)

2	*x*			
3	*x*	*x*		
4	*x*	*x*	*x*	*x*
5		*x*	*x*	*x*
	1	2	3	4

Since both 1 and 5 are final states, there is no subsequent marking and we can stop the iteration.

Indistinguishable pairs : (1, 5)

Distinguishable states : 2, 3, 4

Step 3: Obtain the states of minimized DFA

{1, 5}, 2, 3, 4

Step 4: Identify the start state

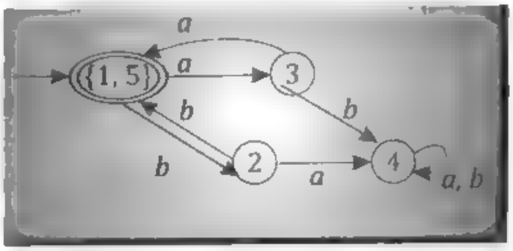
Since 1 is the start state of DFA, the pair {1, 5} is the start state of minimized DFA.

Step 5: Identify the final state

Since 1 and 5 are the final states of DFA, the pair {1, 5} is the final state of minimized DFA.

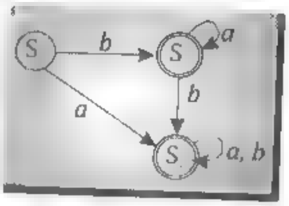
Step 6: Compute transition table

δ	*a*	*b*
→* {1, 5}	3	2
2	4	{1, 5}
3	{1, 5}	4
4	4	4



Problem 12

Minimize the following DFA



Solution

The transition table for the above DFA is given below

δ	a	b
$\rightarrow S_0$	S_2	S_1
$* \; S_1$	S_1	S_2
$* \; S_2$	S_2	S_2

Step 1: Draw the table with initial marking

*	S_1	x	
*	S_1	x	
		S_0	S_1
			*

Both S_1 and S_2 are final states, the is no further marking

Indistinguishable pairs : $\{S_1, S_2\}$

Distinguishable states : S_0

Step 3: Identify the states of minimized

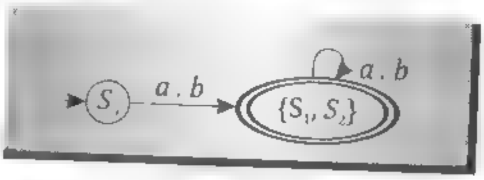
S_0, $\{S_1, S_2\}$

Step 4: S_0 is the start state of minimized DFA.

Step 5: $\{S_1, S_2\}$ is the final state of minimized DFA

Step 6: Compute the transition table

δ	a	b
$\rightarrow S_0$	$\{S_1, S_2\}$	$\{S_1, S_2\}$
$* \; \{S_1, S_2\}$	$\{S_1, S_2\}$	$\{S_1, S_2\}$



PROBLEM 13

Minimize the following DFA

δ	0	1
→ A	B	F
B	G	C
* C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

SOLUTION

The various states of the DFA are A, B, C, D, E, F, G, H.

Step 1: Draw the table with initial markings

	B							
*	C	x	x					
	D			x				
	E			x				
	F			x				
	G			x				
	H			x				
		A	B	C	D	E	F	G
				*				

Step 2: Subsequent marking

δ	a	b
(A, B)	(B, G)	(F, C)
(A, D)	(B, C)	(F, G)
(A, E)	(B, H)	(F, F)
(A, F)	(B, C)	(F, G)
(A, G)	(B, G)	(F, E)
(A, H)	(B, G)	(F, C)
(B, D)	(G, C)	(C, G)
(B, E)	(G, H)	(C, F)

(F, C) is marked so mark (A, B)

(B, C) is marked so mark (A, D)

(B, C) is marked so mark (A, F)

(F, C) is marked so mark (A, H)

(C, G) is marked so mark (B, D)

(C, F) is marked so mark (B, E)

(*B*, *F*)	(*G*, *C*)	(*C*, *G*)	(*C*, *G*) is marked so mark (*B*, *F*)
(*B*, *G*)	(*G*, *G*)	(*C*, *E*)	(*C*, *E*) is marked so mark (*B*, *G*)
(*B*, *H*)	(*G*, *G*)	(*C*, *C*)	
(*D*, *E*)	(*C*, *H*)	(*G*, *F*)	(*C*, *H*) is marked so mark (*D*, *E*)
(*D*, *F*)	(*C*, *C*)	(*G*, *G*)	
(*D*, *G*)	(*C*, *G*)	(*G*, *E*)	(*C*, *G*) is marked so mark (*D*, *G*)
(*D*, *H*)	(*C*, *G*)	(*G*, *C*)	(*C*, *G*) and (*C*, *G*) are marked so mark (*D*, *H*)
(*E*, *F*)	(*H*, *C*)	(*F*, *G*)	(*H*, *C*) is marked so mark (*E*, *F*)
(*E*, *G*)	(*H*, *G*)	(*F*, *E*)	
(*E*, *H*)	(*H*, *G*)	(*F*, *C*)	(*F*, *C*) is marked so mark (*E*, *H*)
(*F*, *G*)	(*C*, *G*)	(*G*, *E*)	(*C*, *G*) is marked so mark (*F*, *G*)
(*F*, *H*)	(*C*, *G*)	(*G*, *C*)	(*C*, *G*) and (*G*, *C*) are marked so mark (*F*, *H*)
(*G*, *H*)	(*G*, *G*)	(*E*, *C*)	(*E*, *C*) is marked so mark (*G*, *H*)

*

B							
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x
	A	*B*	*C*	*D*	*E*	*F*	*G*

*

Repeat step 2:

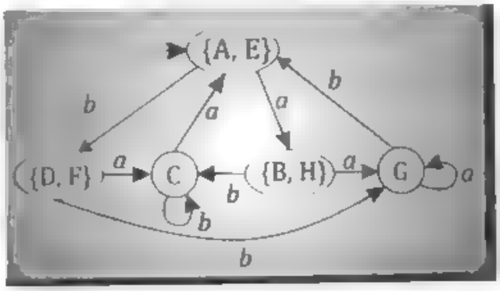
δ	*a*	*b*	
(*A*, *E*)	(*B*, *H*)	(*F*, *F*)	
(*A*, *G*)	(*B*, *G*)	(*F*, *E*)	(*B*, *G*) is marked so mark (*A*, *G*)
(*B*, *H*)	(*G*, *G*)	(*C*, *C*)	
(*D*, *F*)	(*C*, *C*)	(*G*, *G*)	
(*E*, *G*)	(*H*, *G*)	(*F*, *E*)	(*H*, *G*) and (*F*, *E*) are marked so mark (*EG*)

B	x						
• C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G
			*				

Repeat Step 2:

δ	a	b
(A, E)	(B, H)	(F, F)
(B, H)	(G, G)	(C, C)
(D, F)	(C, C)	(G, G)

There is no further marking, so we can stop the iteration.
Indistinguishable pairs: (A, E), (B, H), (D, F)
Distinguishable states: *C* and *G*.



Step 3 : Obtain the states of minimized.....

{A, E}, {B, H}, C, {D, F}, G

Step 4 : Since *A* is the start state of DFA the pair (*A, E*) is the start state of minimized DFA

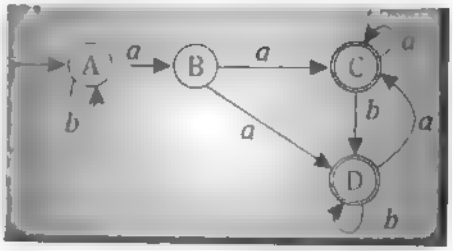
Step 5 : *C* is the final state of DFA, So C is the final state of minimized DFA

Step 6 : Compute transition table

δ	0	1
→ {A, E}	{B, H}	{D, F}
{B, H}	G	C
*C	{A, E}	C
{D, F}	C	G
G	G	{A, E}

Problem 14

Minimize the following DFA



Solution

The transition table of the above DFA

δ	a	b
→ A	B	A
B	C	D
* C	C	D
* D	C	D

The various states of DFA

A, B, C, D

Step 1 : Draw the table with initial marking

	B		
*	C	x	
*	D	x	x	
		A	B	C

*

Step 2 : Subsequent marking

δ	a	b
{A, B}	{B, C}	{A, D}

	B		
*	C	x	
*	D	x	x	
		A	B	C

Indistinguishable pairs : {C, D}

* **Distinguishable states : A, B**

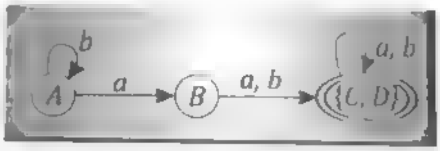
Step 3: Obtain the states of minimized DFA *A, B, {C, D}*

Step 4: A is the start state of minimized DFA

Step 5: Since both C and D are final states of the DFA, the pair $\{C, D\}$ is the final states of minimized DFA

Step 6: Compute transition table

δ	a	b
$\rightarrow A$	B	A
B	$\{C, D\}$	$\{C, D\}$
$* \{C, D\}$	$\{C, D\}$	$\{C, D\}$



– Problem 15 –

Minimize the following DFA

δ	a	b
$\rightarrow 1$	1	2
2	3	2
$*$ 3	1	2

Solution

The various states of the DFA are 1, 2, 3

Step 1: Draw the table with initial marking

	2		
*	3	x	x
		1	2

Step 2: Subsequent marking

δ	a	b	
(1, 2)	(1, 3)	(2, 2)	(1, 3) is marked, so mark (1, 2)

	2	x	
*	3	x	x
		1	2

Since there is no further marking stop the iteration
Indistinguishable pairs : null
Distinguishable states : 1, 2, 3

Since there is no indistinguishable pairs, the given DFA can not be minimized further.

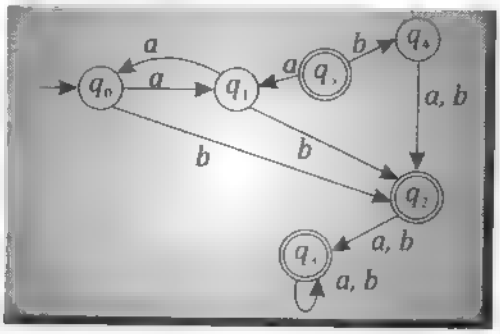
Review Questions

Short Answer Questions

1. Obtain a RE to accept string of a's and b's whose second symbol from the right end is a.
2. Obtain a RE to accept string containing atleast one 0, over $\Sigma = \{0, 1\}$
3. Obtain a RE to accept a string not ending with 001.
4. Write the properties of regular languages.
5. Write the applications of regular languages

Long Answer Questions

1. Define regular expressions and write the basic operations of regular expressions wi examples.
2. Prove that there exists a FA to accept the languages corresponding to a RE.
3. Construct an NFA with ∈-moves for the following regular expressions.
   (i) (0 + 1) (2 + 3)
   (ii) a * b + b * cb
   (iii) ((ab)* + (b + c)*)*
   (iv) (0 + 01)
   (v) ((a * + b*)d*) + k*
   (vi) 0 + 0 * 1
   (vii) (*aa*)*(bb)*b
4. State and prove pumping lemma for regular languages
5. Show that L = {$w|n_a(w) - n_b(w)$} is not regular
6. Show that if L is a regualr languages then the complement of L is also regular
7. Minimize the following DFA



●✱●✱●

Unit 3

CONTEXT FREE GRAMMARS

CHAPTER OUTLINE

A grammar for a computer language is much like that for a natural language ; a set of rules for putting things together. Each grammar corresponds to a language. In this chapter we focus on a special type of grammar called context free grammar. Then we discuss abou push down automata, which is a model of a computer that is more powerful than a finite automata.

3.1 GRAMMAR (G)

In formal language a grammar is a set of production rules. They are also called as language generators, which consist of terminal symbols, non-terminal symbols, a starting symbol and rules. The rules describe how to form strings from the language alphabets that are valid according to the languages syntax. A grammar does not describe the meaning of the strings. The validity of a sentence is determined by the grammar of a language. Each language generated by some grammar can be recognized by some automation.

Definition : Grammer

A grammar is a quadruple, $G = (V, T, P, S)$

Where, V is a finite set of variables or non-terminals

T is a finite set of terminals

P is a finite set of production rules. Each production is of the following form,

$A \rightarrow \alpha$, where

A is a string of symbols from $(V \cup T)^+$

α is a string of symbols from $(V \cup T)^*$

S is the start symbol and $S \in V$.

Note Variables can be replaced by other variables or terminals and it is represented by capital letters.

Terminals cannot be replaced and it is represented using small letters.

3.2 TYPES OF GRAMMAR / CHOMSKY HIERARCHY OF GENERATIVE GRAMMAR

Noam Chomsky who is the founder of formal language theory has classified the grammar into various categories according to the minimal automation required to recognize them. They are:

- Type 0 grammar (Phrase structured/Unrestricted grammar)
- Type 1 grammar (Context Sensitive grammar)
- Type 2 grammar (Context Free grammar)
- Type 3 grammar (Regular grammar)

Type 0 Grammar

A grammar $G = (V, T, P, S)$ is said to be type 0 grammar or unrestricted grammar or phrase structured grammar if all the productions are of the form $\alpha \rightarrow \beta$ where

$$\alpha \in (V \cup T)^* \text{ and } \beta \in (V \cup T)^*$$

In this type of grammar there is no restrictions on length of α and β, but α can not be ϵ.

This is the largest family of grammars which is more powerful than all other types of grammars. Any language can be obtained from this grammar.

The language generated from this grammar is called recursively enumerable language. Only Turing machine can recognize this grammar.

Example:

$$\begin{aligned} S &\rightarrow aAb \mid \epsilon \\ aA &\rightarrow bAA \\ bA &\rightarrow a \end{aligned}$$

Type 1 Grammar

A grammar $G = (V, T, P, S)$ is said to be type 1 grammar or context sensitive grammar if all the productions are of the form $\alpha \rightarrow \beta$

where,

α and $\beta \in (V \cup T)^+$ and $|\beta| \geq |\alpha|$

(ie.,) The length of β

must be atleast as much as the length of α. It is also known as ϵ – Free grammar, since ϵ is not allowed in the left hand side and the right hand side of the production. The language generated by this grammar is context sensitive grammar. Linear Bound Automata (LBA) can be constructed to recognise this language.

Example:

$$\begin{aligned} S &\rightarrow aAb \\ aA &\rightarrow bAA \\ A &\rightarrow b \end{aligned}$$

Type 2 Grammar

A grammar $G = (V, T, P, S)$ is said to be type 2 grammar or context free grammar if all the productions are of the form $A \rightarrow \alpha$ where

$\alpha \in (V \cup T)^*$

and A is a single non-terminal

The symbol ϵ can appear on the right hand side of the production.

The language generated by this grammar is context free language and Pushdown Automata (PDA) can be used to recognise this language.

Example:

$$S \rightarrow aB \mid bB \mid \in$$
$$A \rightarrow aA \mid a$$
$$B \rightarrow bB \mid b \mid \in$$

Type 3 Grammar

The grammar $G = (V, T, P, S)$ is said to be type 3 grammar or Regular grammar or linear grammar if and only if the grammar is right linear or left linear.

A grammar G is said to be right linear if all the productions are of the form

$$A \rightarrow \omega B$$
or
$$A \rightarrow \omega$$

where A, B are variables and $\omega \in T^*$ (i.e.,) w is string of terminals.

Example:

$$S \rightarrow aaB \mid bbA \mid \in$$
$$A \rightarrow aA \mid b$$
$$B \rightarrow bB \mid a \mid \in$$

A grammar G is said to be left linear if all the productions are of the form

$$A \rightarrow B\omega$$
or
$$A \rightarrow \omega$$

where $A, B \in V$ and $\omega \in T^*$

Example:

$$S \rightarrow Baa \mid Abb \mid \in$$
$$A \rightarrow Aa \mid b$$
$$B \rightarrow bB \mid a \mid \in$$

The language generated from the regular grammar is called regular language and these languages are recognised by Finite Automate (FA).

3.3 Context-Free Grammars (CFG)

A context free grammar is a method for (recursively) describing the grammar of a given language. It is a set of variables, each of which represents a language. The language represented by the variables, is described by primitive symbols called terminals. The rules relating to the variables are called productions.

Definition : Conext Free Grammer

A CFG, G is formally defined as a quadruple, $G = (V, T, P, S)$
where V is a finite set of variables or non-terminals.

T is a finite set of terminals

P is a finite set of production rules and

S is a start symbol, $S \in V$.

Each production is of the form $A \to \alpha$,
where A is any single non-terminal and α is any combination of terminals and non-terminals.

Example:

$$\begin{aligned}
&G = (V, T, P, S)\\
\text{Where}\quad &V = \{S, A, B\}\\
&T = \{a, b\}\\
&S = S\\
&P = \{S \to AB\\
&\qquad A \to a\\
&\qquad B \to b\}
\end{aligned}$$

Language Accepted by a CFG

The set of all strings that can be derived from a grammar is said to be the language generated from that grammar.

A language generated by a CFG, G is called a context-free language and is donated by $L(G)$

$$L(G) = \{w \mid w \text{ is in } T^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

e., if w is a terminal string and w can be derived from S, then w is accepted by the grammar.

Note

(1) $A \underset{G}{\Rightarrow} B$ means B can be derived from A by applying only one production.

(2) $A \overset{*}{\underset{G}{\Rightarrow}} B$ means B can be derived from A by applying more than one production.

Example:

Consider the following grammar, $G = (V, T, P, S)$

$$\begin{aligned}
\text{Where}\quad &V = \{S, A, B\}\\
&T = \{a, b\}\\
&S = S\\
&P = \{S \to AB\\
&\qquad A \to a\\
&\qquad B \to b\}
\end{aligned}$$

Here S produces AB, and we can replace A by a, and B by b. The only accepted string is ab

i.e., $S \rightarrow AB$ $\quad\quad \rightarrow a\,B\,A \rightarrow a$

$\rightarrow a\,b\,B \rightarrow b$

$\therefore L(G) = a\,b$

Type	Machine	Grammar
III	Finite Automata	Regular
II	Push Down Automata	Context-Free
I	Linear Bound Automata	Context Sensitive
0	Turing Machine	Recursive

3.4 Design of Language from CFG

Problem 1

Consider a grammar $G = (V, T, P, S)$

Where, $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow a\,S \mid b\}$

Find the language accepted by G.

Solution

$S \rightarrow a\,S$ $\quad\quad S \rightarrow b \quad S \rightarrow a\,S$

$\rightarrow a\,a\,S$ $\quad\quad\quad\quad \rightarrow a\,b$

$\rightarrow a\,a\,a\,S$

$\rightarrow a\,a\,a\_\_\_\_\_\_a\,S$

$\rightarrow a\,a\,a\_\_\_\_\_\_a\,b$

i.e., $S \rightarrow a^*b$

$\therefore$ $L(G) = a^*b$

Problem 2

Consider the following grammar $G = (V, T, P, S)$

where, $V = \{S\}$

$T = \{a\}$

$S = S$

$P = \{S \rightarrow a\,S \mid \in\}$

Find the language accepted by G.

Solution

$$\begin{array}{llll} S \to \in & S \to aS & S \to aS & S \to aS \\ & \to a\in & \to aaS & \to aaS \\ & \to a & \to aa\in & \to aa\_\_\_aS \\ & & \to aa & \to aa\_\_\_aa \end{array}$$

i.e., $S \to a^* \in$

$\to a^*$

$\therefore$ $L(G) = a^*$

Problem 3

Find the language accepted by the following grammar $G = (V, T, P, S)$

where, $V = \{S\}$

$T = \{a\}$

$S = S$

$P = \{S \to aaS \mid aa\}$

Solution

$$\begin{array}{lll} S \to aa & S \to aaS & S \to aaS \\ & \to aaaa & \to aaaaS \\ & & \to aaaa\_\_\_aaS \\ & & \to aaaa\_\_\_aaaa \end{array}$$

i.e., $S \to (aa)^+$

$\therefore$ $L(G) = (aa)^+$

Problem 4

Find the language accepted by the following grammar.

$G = (\{S, T\}, \{a, b, c\}, P, S\})$

where, $P = \{S \to aT$

$T \to bbT \mid c\}$

Solution

$$\begin{array}{lll} S \to aT & S \to aT & S \to aT \\ \to ac & \to abbT & \to abbT \\ & \to abbc & \to abbbbT \\ & & \to abb\_\_\_\_bbc \end{array}$$

i.e., $S \to a(bb)^* c$

$\therefore$ $L(G) = a(aa)^* c$

Example 5

Consider the following grammar

$S \rightarrow a\,C\,a$

$C \rightarrow a\,C\,a|b$

Write the language generated by the grammar

SOLUTION

$S \rightarrow a\,C\,a$	$S \rightarrow a\,C\,a$	$S \rightarrow a\,C\,a$
$\rightarrow a\,b\,a$	$\rightarrow a\,a\,C\,a\,a$	$\rightarrow a\,a\,C\,a\,a$
	$\rightarrow a\,a\,b\,a\,a$	$\rightarrow a\,a\,\_\_\_C\_\_\,a\,a$
		$\rightarrow a\,a\,a\,\_\,b\,\_\_\,.\,a\,a\,a$

i.e., $S \rightarrow a^n\,b\,a^n$

$\therefore$ $L\,(G) = \{a^n\,b\,a^n \mid n \geq 1\}$

3.5 Design of Context free Grammar from Finite Automata

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata accepting L, A grammar $G = (V, T, P, S)$ can be constructed where,

$V = \{q_0, q_1 \text{---} q_n\}$

i.e., the states of DFA will be the variables in the grammar.

$T = \Sigma$

i.e., the input alphabets of DFA are the terminals in grammar.

$S = q_0$

i.e., the start state of DFA is the start symbol in the grammar.

The productions P from the transitions can be obtained as

(1) if $\delta\,(q_i, a) = q_j$

then introduce the transition

$q_i \rightarrow a\,q_j$

(2) if $q \in F$

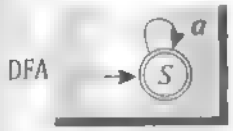
i.e , if q is the final state in FA, then introduce the production.

$q \rightarrow \in$

Problem 1

btain grammar to generate string consisting of any number of a's and write the nguage generated by the grammar.

Solution



Transitions	Grammar
$\delta (S, a) = S$	$\therefore S \rightarrow a S$
S is a final state	$\therefore S \rightarrow \in$

. the grammar $G = (V, T, P, S)$ where

$V = \{ S \}$

$T = \{ a \}$

$S = S$

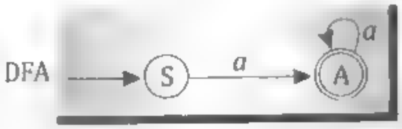
$P = \{ S \rightarrow a S | \in \}$

he language generated by the grammar, $L (G) = \{a^n \mid n \geq 0\}$

Problem 2

Obtain grammar to generate string consisting of atleast one a.

Solution



Transitions	Grammar
$\delta (S, a) = A$	$\therefore S \rightarrow a A$
$\delta (A, a) = A$	$\therefore A \rightarrow a A$
A is a final state	$\therefore A \rightarrow \in$

$\therefore$ the grammar, $G = (V, T, P, S)$

where $V = \{ S, A \}$

$T = \{ a \}$

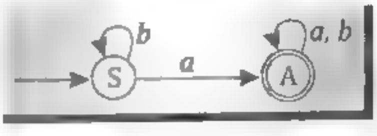
$S = S$

$P = \{ S \rightarrow a A$

$A \rightarrow a A \mid \in \}$

Problem 3

Obtain a grammar for the following DFA



Solution

Transitions	Grammar
$\delta(S, a) = A$	$\therefore S \rightarrow aA$
$\delta(S, b) = S$	$\therefore S \rightarrow bS$
$\delta(A, a) = A$	$\therefore A \rightarrow aA$
$\delta(A, b) = A$	$\therefore A \rightarrow bA$

A is a final state $\therefore A \rightarrow \in$

$\therefore$ **the grammar, $G = (V, T, P, S)$**

where $V = \{S, A\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow aA \mid bS$

$A \rightarrow aA \mid bA \mid \in\}$

Problem 4

Obtain grammar to generate string consisting of any number of a's and b's wi atleast one a.

Solution

Transitions	Grammar
$\delta(S, a) = A$	$\therefore S \rightarrow aA$
$\delta(S, b) = S$	$\therefore S \rightarrow bS$
$\delta(A, a) = A$	$\therefore A \rightarrow aA$
$\delta(A, b) = A$	$\therefore A \rightarrow bA$

A is a final state $\therefore A \rightarrow \in$

$\therefore$ **the grammar, $G = (V, T, P, S)$**

where $V = \{S, A\}$

$T = \{a, b\}$

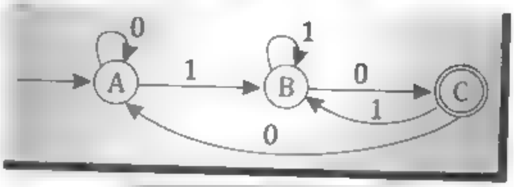
$S = S$

$P = \{S \rightarrow aA \mid bS$

$A \rightarrow aA \mid bA \mid \in\}$

Problem 5

Obtain a grammar for the following DFA



Solution

Transitions	Grammar
$\delta(A, 0) = A$	$\therefore A \rightarrow 0A$
$\delta(A, 1) = B$	$\therefore A \rightarrow 1B$
$\delta(B, 0) = C$	$\therefore B \rightarrow 0C$
$\delta(B, 1) = B$	$\therefore B \rightarrow 1B$
$\delta(C, 0) = A$	$\therefore C \rightarrow 0A$
$\delta(C, 1) = B$	$\therefore C \rightarrow 1B$

C is the final state $\therefore C \rightarrow \in$

∴ **the grammar,** $G = (V, T, P, S)$

where $V = \{A, B, C\}$

$T = \{0, 1\}$

$S = A$

$P = \{A \rightarrow 0A \mid 1B$

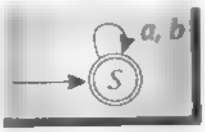
$B \rightarrow 0C \mid 1B$

$C \rightarrow 0A \mid 1B \mid \in\}$

PROBLEM 6

Obtain a grammar to generate string consisting of any number of *a*'s and *b*'s.

SOLUTION

DFA 

Transitions	**Grammar**
$\delta(S, a) = S$	$\therefore S \rightarrow aS$
$\delta(S, b) = S$	$\therefore S \rightarrow bS$

S is a final state

∴ $S \rightarrow \in$.

The equivalent grammar $G = (V, T, P, S)$

where $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow aS \mid bS \mid \in\}$

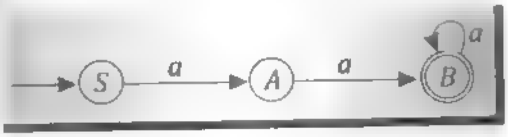
The language generated by this grammar, $L(G) = \{(a + b)^n \mid n \geq 0\}$

PROBLEM 7

Obtain grammar to generate string consisting of at least two a's.

SOLUTION

DFA



Transitions	**Grammar**
$\delta(S, a) = A$	$\therefore S \rightarrow aA$
$\delta(A, a) = B$	$\therefore A \rightarrow aB$
$\delta(B, a) = B$	$\therefore B \rightarrow aB$
B is a final state	$\therefore B \rightarrow \in$.

The equivalent grammar $G = (V, T, P, S)$

where, $V = \{S, A, B\}$

$T = \{a\}$

$S = \{S\}$

$P = \{S \rightarrow a\,A; \quad A \rightarrow a\,B$

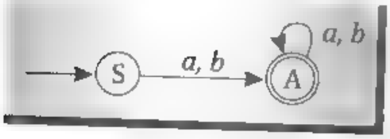
$B \rightarrow a\,B \mid \in \}$

Problem 8

Obtain grammar to generate string consisting of any number of a's and b's atleast one a or atleast one b.

Solution

DFA



Transitions	Grammar
$\delta(S, a) = A$	$\therefore S \rightarrow a\,A$
$\delta(S, b) = A$	$\therefore S \rightarrow b\,A$
$\delta(A, a) = A$	$\therefore A \rightarrow a\,A$
$\delta(A, b) = A$	$\therefore A \rightarrow b\,A$
A is a final state	$\therefore A \rightarrow \in$

The equivalent grammar $G = (V, T, P, S)$

where $V = \{S, A\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow a\,A \mid \underline{b\,A}$

$A \rightarrow a\,B \mid b\,A \mid \in \}$

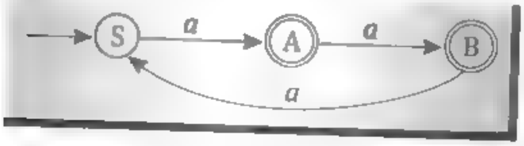
Problem 9

Obtain grammar to accept the following language

$L = \{ w \mid |w| \bmod 3 > 0, \text{ where } w \in \{a\}^* \}$

Solution

The minimum string that can be accepted is either a or aa. So the DFA is



Transitions	Grammar
$\delta(S, a) = A$	$\therefore S \rightarrow a\,A$
$\delta(A, a) = B$	$\therefore A \rightarrow a\,B$
$\delta(B, a) = S$	$\therefore B \rightarrow a\,S$
A is a final state	$\therefore A \rightarrow \in$
B is the final state	$\therefore B \rightarrow \in$

The equivalent grammar, $G = (V, T, P, S)$

where $V = \{ S, A, B \}$

$T = \{ a \}$

$S = S$

$P = \{ S \rightarrow a A$

$A \rightarrow a B \mid \in$

$B \rightarrow a S \mid \in \}$

3.6 Design of Context free Grammar from Languages

.n the previous problems context free grammars were formed for simple languages using finite automata. Here we are designing the CFG from the language directly, without using finite automata. The steps are as follows.

Step 1: Understand the language specification by listing the examples of the strings in the language.

Step 2: Determine the base case productions of CFG by listing the shortest strings in the language.

Step 3: Identify the rules for combining smaller sentences into larger ones.

Step 4: Test the CFG obtained on a number of carefully chosen examples. All of the base cases should be tested, along with all of the alternative production. Also, check whether the grammar is consistent with the strings listed in Step 1.

Problem 1

Obtain a CFG for the following language. $L = a^n b^n, n \geq 0$

Solution

When $n = 0$, the production is $S \rightarrow \in$

When $n = 1$, the string is ab, i.e., for every a one b has to be generated

$\therefore$ The production is $S \rightarrow a S b$.

The derivation for the string $a\,b$ and $aa\,bb$ is as shown below

$S \rightarrow a S b$

$\rightarrow a \in b$

$\rightarrow a \quad b$

$S \rightarrow a S b$

$\rightarrow a a S b b$

$\rightarrow a a \in b b$

$\rightarrow a a b b$

$\therefore$ The CFG, $G = (V, T, P, S)$

where $V = \{ S \}$

$T = \{ a, b \}$

$S = S$

$P = \{ S \rightarrow a S b \mid \in \}$

Problem 2

Obtain a CFG for the following language, $L = \{ a^n b^n \mid n \geq 1 \}$

Solution

When $n = 1$, the string is $a\,b$. ∴ the production is $S \to a\,b$

When n is more than one, the string is *aabb, aaabbb,* _ _ _

∴ The production is $S \to a\,S\,b$

The derivation for the string *ab* and *aaa bbb* is as shown below

$S \to a\,b$

$$\begin{aligned} S &\to a\,S\,b \\ &\to a\,a\,S\,b\,b \\ &\to a\,a\,a\,b\,b\,b \end{aligned}$$

∴ The CFG, $G = (V, T, P, S)$

Where $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \{S \to a\,S\,b \mid ab\}$

Problem 3

Let $\Sigma = \{a, b\}$, obtain a CFG for the set of all palindromes over Σ.

Solution

$\in$ is palindrome	∴ the production is $S \to \in$
a|b is palindrome	∴ the production is $S \to a\|b$
If w is palindrome then a w a and b w b are palindromes	∴ the production is $S \to a\,S\,b \mid b\,S\,b$

Now with these productions we can generate the palindromes a b a or abababab a as shown below.

$$\begin{aligned} S &\to a\,S\,a \\ &\to a\,b\,a \end{aligned}$$

$$\begin{aligned} S &\to a\,S\,b \\ &\to a\,b\,S\,b\,a \\ &\to a\,b\,a\,S\,a\,b\,a \\ &\to a\,b\,a\,b\,S\,b\,a\,b\,a \\ &\to a\,b\,a\,b\,a\,b\,a\,b\,a \\ &\to a\,b\,a\,b\,a\,b\,a\,b\,a \end{aligned}$$

∴ The CFG, $G = (V, T, P, S)$

Where $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow aSa \mid bSb \mid a \mid b \mid \in\}$

Problem 4

Obtain a CFG for the following language. $L = \{ww^R \mid \text{where } w \in \{a, b\}^*\}$

Solution

The strings that can be generated from this language are ∈, aa, bb, abba, babbab,

∴ The productions are $S \rightarrow \in$

$S \rightarrow aSa \mid bSb$

The derivation for the string abba is as shown below

$S \rightarrow aSa$

$\rightarrow abSba$

$\rightarrow ab\in ba$

$\rightarrow abba$

∴ The CFG, $G = (V, T, P, S)$

Where $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow aSb \mid bSb \mid \in\}$

Problem 5

Obtain a CFG for the following language $L = \{wcw^R \mid \text{where } w \in \{a, b\}^*\}$

Solution

The strings that can be generated from this language are, c, aca, bcb, abcba, _ _ _ _

∴ the productions are

$S \rightarrow c$

$S \rightarrow aSa \mid bSb$

The CFG, $G = (V, T, P, S)$

where $V = \{S\}$

$T = \{a, b, c\}$

$S = S$

$P = \{S \rightarrow aSa \mid bSb \mid c\}$

Problem 6

Obtain a CFG for the following language $L = \{ w \mid n_a(w) = n_b(w) \}$

Solution

Number of a's in w should be equal to the number of b's in w. The strings that can be generated are, $\in$, *aa, bb, abab, bbbaabbb* etc.

For zero a's and b's, the the production is $S \rightarrow \in$.

a can be followed by b $\therefore$ the production is $S \rightarrow a\,S\,b$.

b can be followed by a $\therefore$ the production is $S \rightarrow b\,S\,a$.

For the string *abba* the production is $S \rightarrow SS$

$\therefore$ the CFG, $G = (V, T, P, S)$

where $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow S\,S \mid a\,S\,b \mid b\,S\,a \mid \in\}$

Problem 7

Obtain a grammar to generate set of all strings with atleast one a, when $\Sigma = \{a, b\}$

Solution

The language contains one or more a's and any number of b's. The strings generated from this grammar are

a, ab, aaab, bab, ____

For any number of b's, the production is $S \rightarrow b\,S$.

For atleast one a from S, the production if $S \rightarrow a\,A$.

For a followed by any number of a's and b's, the productions are

$A \rightarrow \in$

$A \rightarrow a\,A$

$A \rightarrow b\,A$

$\therefore$ The grammar, $G = (V, T, P, S)$

where $V = \{S, A\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow a\,A \mid b\,S$

$A \rightarrow a\,A \mid b\,A \mid \in\}$

Problem 8

Obtain a grammar to generate the language $L = \{ a^n b^m \mid n \geq 0, m > n \}$

Solution

If $n = 0$, and $m \geq 1$the string will be bb^*.

If $n = 1$ and $m \geq 2$, the string will be $abbb^*$

If $n = 2$ and $m \geq 3$, the string will be $aabbbb^*$ etc.,

∴ The productions are

$S \rightarrow AB$

$A \rightarrow \in$ [When $n = 0$]

$A \rightarrow a\,A\,b$ [When $n = 1$ two b's, $n = 2$ three b's etc.,]

$B \rightarrow b \mid bB$

∴ the grammar **$G = (V, T, P, S)$**

where $V = \{S, A, B\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow AB$

$A \rightarrow a\,A\,b \mid \in$

$B \rightarrow b\,B \mid b\,\}$

Problem 9

Obtain a grammar to generate the language $L = \{ a^m b^m c^n \mid m \geq 1$ and $n \geq 0\}$

Solution

If $m = 1$ and $n = 0$ the string will be ab.

If m = 1 and $n = 1$ the string will be abc.

If m = 1 and $n = 1$ the string will be $aabbcc$.

∴ the productions are

$S \rightarrow A\,C$

$A \rightarrow ab \mid a\,A\,b$ [For every a equivalent b should be there]

$C \rightarrow c\,C \mid \in$

the grammar **$G = (V, T, P, S)$**

where $V = \{S, A, C\}$

$T = \{a, b, c\}$

$S = S$

$P = \{S \rightarrow A\,C$

$A \rightarrow a\,A\,b \mid a\,b$

$C \rightarrow c\,C \mid \in\}$

Problem 10

Obtain a CFG for the language of even palindrome, over the alphabet $\Sigma = \{a, b\}$

Solution

Strings of even palindromes are *abba, aabbaa, baba abab,* _ _ _ .

The CFG, $G = (V, T, P, S)$

where $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow a\,S\,a \mid b\,S\,b \mid \in\}$

Problem 11

Obtain a CFG for the set of all well formed parentheses.

Solution

- $\in$ and ( ) are balanced ∴ the production is S → ∈ | ( )
- If *w* is balanced, then (*w*) is also balanced. ∴ the production is S → (S)
- If *w* and *x* are balanced, then wx is also balanced. ∴ the production is S → S S

∴ The CFG, $G = (V, T, P, S)$

where $V = \{S$

$T = \{(\,)\}$

$S = S$

$P = \{S \rightarrow SS \mid (S) \mid (\,) \mid \in\}$

Problem 12

Give CFG for the language $L = \{ x \in \{0, 1\}^* \mid x$ has unequal number of 0's and 1's $\}$

Solution

The CFG, $G = (V, T, P, S)$

where $V = \{S, A, B\}$

$T = \{0, 1\}$

$S = S$

$P = \{ S \rightarrow A \mid B$

$A \rightarrow 0 \mid 0A \mid 1AA \mid A1A \mid AA1$

$B \rightarrow 1 \mid 1B \mid 0BB \mid B0B \mid BB0\}$

Problem 13

Obtain a grammar to generate the set of all strings with exactly one a, over $\Sigma = \{a, b\}$

Solution

The grammar $G = (V, T, P, S)$

where $V = \{S, A\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow aA \mid bS$

$A \rightarrow bA \mid \in\}$

Problem 14

Obtain a grammar to generate the set of all strings with atleast one a over $\Sigma = \{a, b\}$

Solution

The grammar $G = (V, T, P, S)$

where $V = \{S, A\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow aA \mid bS$

$A \rightarrow aA \mid bA \mid \in\}$

3.7 - Design of CFG from Regular Expressions

Problem 1

Obtain the CFG for the regular expression $(a + b)^* ab (a + b)^*$

Solution

The given regular expression represents strings of a's and b's having a substring ab (a+b)* represents strings of a's and b's including NULL string.

$\therefore$ The CFG, $G = (V, T, P, S)$

where $V = \{S, A\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow A\,ab\,A$

$A \rightarrow aA \mid bA \mid \in\}$

Problem 2

Obtain the CFG for the regular expression $(a + b)^* ab$

Solution

The RE represents strings of a's and b's ending with ab. $(a + b)^*$ represents strings of a's and b's including NULL string

$\therefore$ the grammar, $G = (V, T, P, S)$

where $V = \{S, A\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow A\,ab\,A$

$A \rightarrow aA \mid bA \mid \in\}$

Problem 3

Obtain the CFG for the Regular Expression (011+1)* (01)*

Solution

The given RE is of the form AB, where A is 011 or 1 or NULL string and B is 01 or NULL string

$\therefore$ the CFG, $G = (V, T, P, S)$

where $V =$

$T = \{0, 1\}$

$S = S$

$P = \{S \rightarrow A\,B$

$A \rightarrow 011\,A \mid 1\,A \mid \in$

$B \rightarrow 01\,B \mid \in\}$

3.8 Parsing / Derivation

The sequence of substitutions used to obtain a string, is called derivation or parsing. It means replacing an instance of a given strings non-terminal, by the right hand side of the production, whose left hand side contains the non-terminal to be replaced.

For example consider the following productions,

$S \rightarrow x \alpha y$

$\alpha \rightarrow \beta$

Then the replacement of α by β in x α y is called parsing or derivation.

$S \rightarrow x \alpha y$

$\rightarrow x \beta y$

Parsing produces a new string from a given string. Therefore derivation can be used repeatedly to obtain a new string from a given string. If the string obtained after parsing contains only terminal symbols, then no further derivations are possible.

Forms of Derivation

There are two types of derivation

i) Left most derivation, and

ii) Right most derivation

Left most Derivation

Definition : Left Most Derivation

A derivation $A \overset{*}{\Rightarrow} w$ is called left most derivation if we apply a production only to the left most variable at every step.

Example:

Consider the following production's

$S \rightarrow XYX$

$X \rightarrow a$

$Y \rightarrow b$

The left most derivation for the string aba is as follows.

$S \rightarrow XYX$

$\rightarrow aYX \quad [\because X \rightarrow a]$

$\rightarrow abX \quad [\because Y \rightarrow b]$

$\rightarrow aba \quad [\because X \rightarrow a]$

Note In left most derivation, always the left most non terminal is replaced.

Right Most Derivation

Definition : Right Most Derivation

A derivation $A \overset{*}{\Longrightarrow} w$ is a right most derivation if we apply production to the rightmost variable at every step.

Example:

Consider the following productions

$S \to XYX$

$X \to a$

$Y \to b$

The right most derivation for the string aba is as follows.

$S \to XYX$

$\to XYa \qquad [\because X \to a]$

$\to Xba \qquad [\because Y \to b]$

$\to aba \qquad [\because X \to a]$

Note In right most derivation always the rightmost non terminal is replaced.

PROBLEM 1

For the following production

$S \to AB$

$A \to aaA \mid \epsilon$

$B \to Bb \mid \epsilon$

write the left most and right most derivation for the string aab.

SOLUTION

Left most derivation for *aab*

$S \to AB$

$\to aaAB$

$\to aaB$

$\to aaBb$

$\to aab$

Right most derivation for *aab*

$S \to AB$

$\to ABb$

$\to Ab$

$\to aaAb$

$\to aab$

Problem 2

Find the left most and right most derivation for the string *aaabba bbba*, where *G* is

{S → a B | b A

A → a | a S | b A A

B → b | b S | a B B}

Solution

Left most derivation for *aaabb abbba*

S → a B

→ aa BB

→ aaa BBB

→ aaa bBB

→ aaa bbB

→ aaa bb aBB

→ aaa bba bB

→ aaa bb abbS

→ aaa bba bbbA

→ aaa bba abbba

Right most derivation for *aaa bba bbba*

S → a B

→ aa BB

→ aa Ba BB

→ aa Ba BbS

→ aa Ba BbbA

→ aa Ba Bbba

→ aa Ba bbba

→ aaa BBa bbba

→ aaa Bba bbba

→ aaa bba bbba

Problem 3

Write the left most and right most derivation for the string *abbbb* using the production.

S → a A B

A → b B b

B → A | ∈

Solution

Left most derivation for *abbbb*

S → a A B

→ a b B b B

→ a b A b B

→ a b b B b b B

→ a bb bb B

→ a bb bb

Right most derivation for abbbb

S → a A B

→ a A

→ a b B b

→ a b A b

→ a b b B b b

→ a bbbb

3.9 Parse Tree / Derivation Tree (PT)

The derivations or passing process in a context free grammar can be represented usin trees. Such trees are called passe tree/derivation tree. The vertices of parse tree are labele with terminals or non terminals. The root of the tree is the start variable, all internal nodes are labeled with variables and the leaves are labeled with terminals. The children of a node are labeled from left to right with the right hand side of the production used. If an interio vertex is labeled A and the branches are labelled as $x_1, x_2 --- x_n$ then $A \rightarrow x_1, x_2 --- x_n$ mus be a production in P.

Definition : Parse Tree

A derivation tree/parse tree for a CFG, G = (V, T, P, S) is a tree satisfying the following conditions

(i) Every vertex has a label which is a variable or terminal or $\in$.

(ii) The label of the root is S.

(iii) The label of an internal vertex is a variable.

(iv) If a vertex has label A and x_1, x_2 x_n are children of A then $A \rightarrow x_1, x_2 --- x_n$ must be a production in P.

(v) If vertex n has label $\in$, then n is a leaf node and is the only son of its father.

Example:

Let the CFG, $G = (V, T, P, S)$

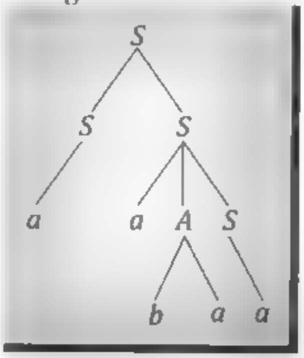
where $V = \{S, A\}$

$T = \{a, b\}$

$P = \{S \rightarrow SS \mid aAS \mid a$

$A \rightarrow SbA \mid ba\}$

The parse tree for the string *aabaa* is given as follows.



SUBTREE

A subtree of a parse tree is a particular vertex at the tree together with all the vertices and edges connecting them. It looks like a parse tree, except the root. The root may not be the start symbol of the grammar.

Definition : Subtree

A subtree of a parse tree, is a tree

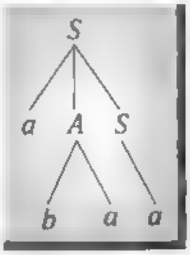
(i) whose root is some vertex v of T.

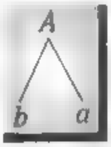
(ii) whose vertices are the descendants of v together with their labels and

(iii) whose edges are those connecting the descendants of V.

Example:

For the parse tree of the previous example, the subtrees are





Problem 1

For the grammar G – with production rules,

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

where $V = \{ E \}$, $T = \{ id \}$, $S = \{ E \}$, obtain the derivation and the parse tree for the string $w = id + id * id$

Solution

Derivation for *w*:

$E \rightarrow E + E$

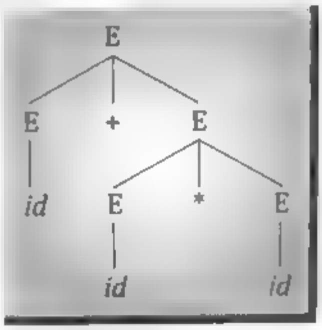
$\rightarrow E + E * E$

$\rightarrow E + E * id$

$\rightarrow E + id * id$

$\rightarrow id + id * id$

Parse tree for id + id * id



PROBLEM 2

For $G = \{\{S, B\}, \{c, d\}, P, S\}$

where $P = \{S \rightarrow c\,B\,S \mid c$

$B \rightarrow S\,d\,B \mid S\,S \mid d\,c\}$

Find the left most derivation and draw the parse tree.

SOLUTION

Left most Derivation

$S \rightarrow cBS$

$\rightarrow cSdBS$

$\rightarrow ccdBS$

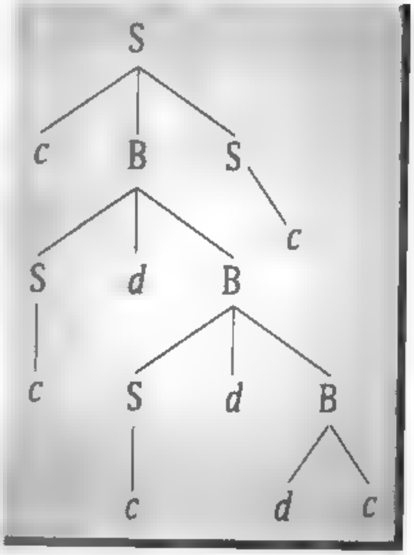
$\rightarrow ccdSdBS$

$\rightarrow ccdcdBS$

$\rightarrow ccdcddcS$

$\rightarrow ccdcddcc$

Parse tree for *ccdcddcc*



PROBLEM 3

Let G be a grammar with P given by:

$S \rightarrow aB \mid bA$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

For the string $w = aaabbabbba$, find the left and right most derivations and also draw the parse tree.

Solution

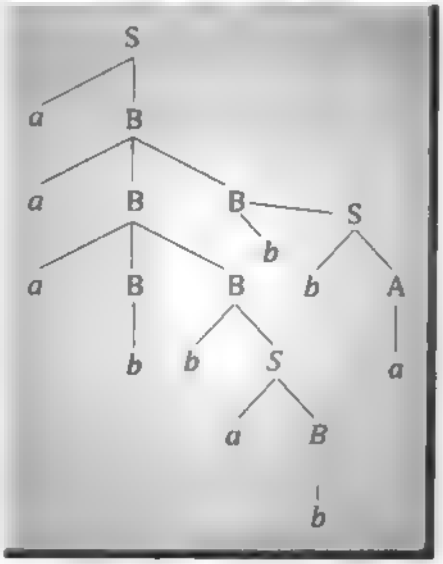
Left most derivation for w :

$S \rightarrow aB$
$\rightarrow aaBB$
$\rightarrow aaaBBB$
$\rightarrow aaabBB$
$\rightarrow aaabbSB$
$\rightarrow aaabbaBB$
$\rightarrow aaabbabB$
$\rightarrow aaabbabbS$
$\rightarrow aaabbabbbA$
$\rightarrow aaabbabbba$

Right most derivation for w :

$S \rightarrow aB$
$\rightarrow aaBB$
$\rightarrow aaBbS$
$\rightarrow aaBbbA$
$\rightarrow aaBbba$
$\rightarrow aaaBBbba$
$\rightarrow aaaBbbba$
$\rightarrow aaabSbbba$
$\rightarrow aaabSbbba$
$\rightarrow aaabbAbbba$
$\rightarrow aaabbabbba$

Parse tree for right most derivation of w.



PROBLEM 4

For $G = (\{E, I\}, \{a, b, c + *\}, E, P\}$

where $P = \{E \rightarrow I$

$E \rightarrow E + E$

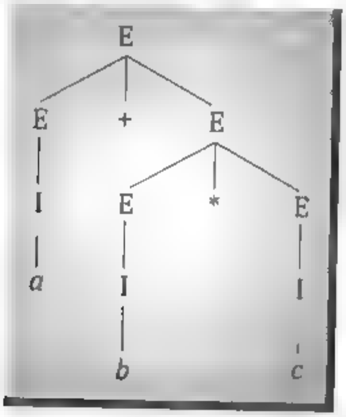
$E \rightarrow E * E$

$I \rightarrow a|b|c\}$

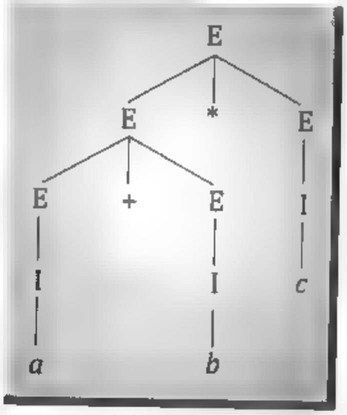
obtain two derivation trees for $a + b * c$

SOLUTION

First derivation tree for $a + b * c$



Second derivation tree for $a + b * c$



PROBLEM 5

Derive the string 0111 00 from the following grammar, $S \to 0\,S\,1\,S \mid 1\,S\,0\,S \mid \epsilon$ and draw the parse tree.

SOLUTION

Derivation for 0111 00

$S \to 0\,S\,1\,S$

$\to 0\,1\,S$

$\to 0\,1\,1\,S\,0\,S$

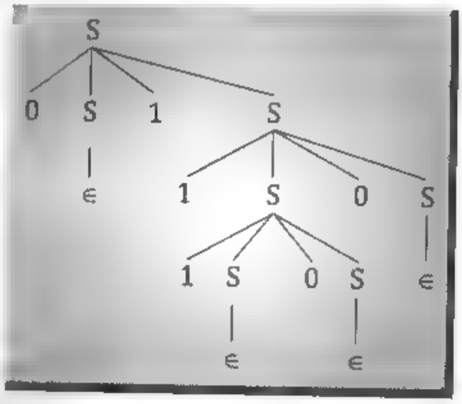
$\to 0\,1\,1\,1\,S\,0\,S\,0\,S$

$\to 0\,1\,1\,1\,0\,S\,0\,S$

$\to 0\,1\,1\,1\,0\,0\,S$

$\to 0\,1\,1\,1\,0\,0$

parse tree for 0 1 1 1 0 0



PROBLEM 6

Derive the string *abaaba* from the following grammar,

$S \to a\,S\,a$

$S \to b\,S\,b$

$S \to a \mid b \mid \epsilon$

and draw the parse tree.

SOLUTION

Derivation for *abaaba*

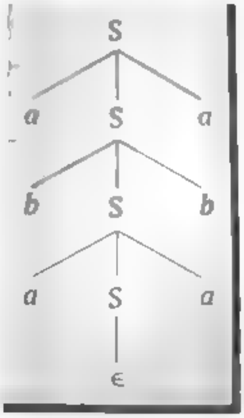
$S \rightarrow a\,S\,a$

$\rightarrow a\,b\,S\,b\,a$

$\rightarrow a\,b\,a\,S\,a\,b\,a$

$\rightarrow a\,b\,a\,a\,b\,a$

Parse tree for *a b a a b a*



3.10 APPLICATION OF CFG

ontext free grammars are the most commonly used kind of grammar in computer science. They provide a more powerful mechanism for language specification. Some of the applications of CFG are

- Used in digital design
- Used as basis for compiler design and implementation
- Specifying syntax of programming languages
- Representing syntactic structures in natural languages

3.11 AMBIGUOUS GRAMMAR

Definition : Ambiguous Grammar

An ambiguous grammar is a context free grammar for which there exists a string that can have more than one leftmost derivation, while an unambiguous grammar is a context free grammar for which every valid string has a unique left most derivation.

Let G = (V, T, P, S) be a context free grammar. A grammar G is ambiguous if and only if there exists at east one string w ∈ T* for which two or more different parse tree exist by applying either the left most derivation or right most derivation.

— PROBLEM 1 —

Show that the following grammar is ambiguous

$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow E * E$

$E \to E \mid E$

$E \to (E)$

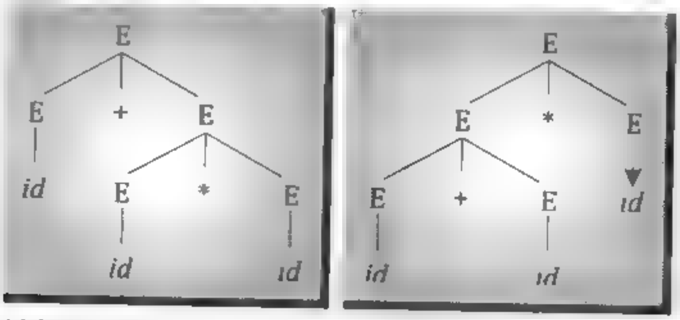
$E \to id$

Solution

The string $id + id * id$ can be obtained from left most derivation in two ways as shown below

$E \to E + E$	$E \to E * E$
$\to id + E$	$\to E + E * E$
$\to id + E * E$	$\to id + E * E$
$\to id + id * E$	$\to id + id * E$
$\to id + id * id$	$\to id + id * id$

The corresponding parse trees for the two left most derivations are shown below.



For the string $id + id * id$, by applying left most derivations we get two different parse trees
∴ the grammar is ambiguous.

Problem 2

Is the following grammar ambiguous?

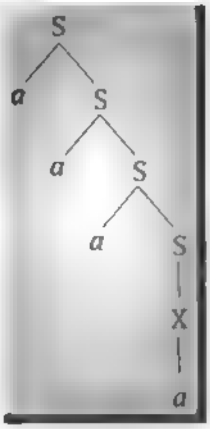
$S \to a S \mid x$ $\qquad$ $X \to a X \mid a$

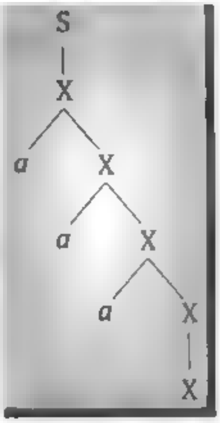
Solution

Consider the two left most derivations for the string *aaaa*.

$S \to a s$	$S \to x$
$\to a\,as$	$\to a x$
$\to a a a s$	$\to a a x$
$\to a a a x$	$\to a a a x$
$\to a a a a$	$\to a a a a$

The corresponding derivation trees for the two leftmost derivations are shown below





Since the two parse trees are different for the same string *aaaa*, the grammar is ambiguous.

PROBLEM 3

Is the following grammar ambiguous?

S* → *a B* | *b A

A* → *a S* | *b A A* | *a

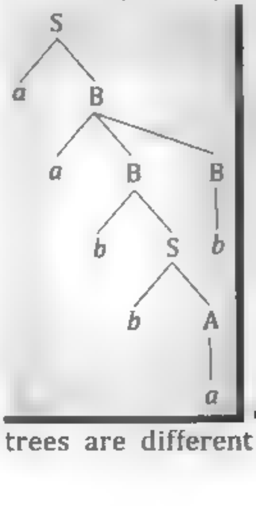
B* → *b S* | *a B B* | *b

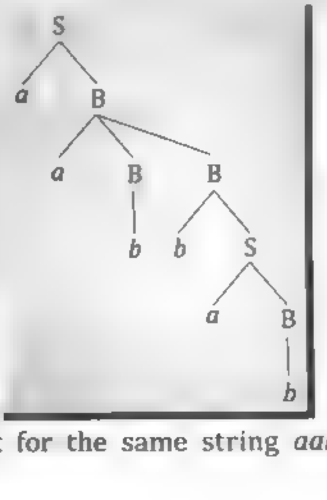
SOLUTION

The string *aabbab* can be obtained from left most derivation in two ways.

S → a B	S → a B
→ a a B B	→ a a B B
→ a a b S B	→ a a b B
→ a a b b A B	→ a a b b S
→ a a b b a B	→ a a b b a B
→ a a b b a b	→ a a b b a b

The corresponding parse trees are shown below





Since the two parse trees are different for the same string *aabbab* the grammar is ambiguous.

Problem 4

Show that the following grammar is ambiguous.

$S \rightarrow aSbS$

$S \rightarrow bSaS$

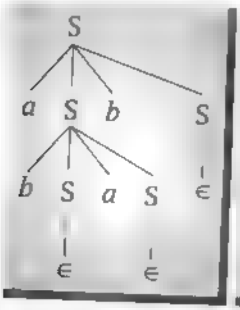
$S \rightarrow \Sigma$

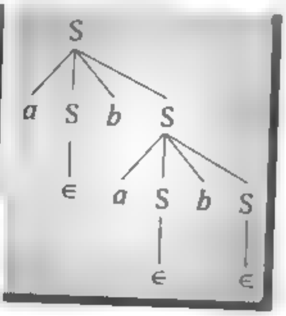
Solution

The string *abab* can be obtained from left most derivation in two ways as shown below.

$S \rightarrow aSbS$	$S \rightarrow aSbS$
$\rightarrow abSaSbS$	$\rightarrow aSbaSbS$
$\rightarrow abaSbS$	$\rightarrow abaSbS$
$\rightarrow ababS$	$\rightarrow ababS$
$\rightarrow abab$	$\rightarrow abab$

The corresponding parse tree for the two left most derivations are shown below.





Since the two parse trees are different for the same string abab, the grammer is ambiguous.

Problem 5

Is the following grammar ambiguous?

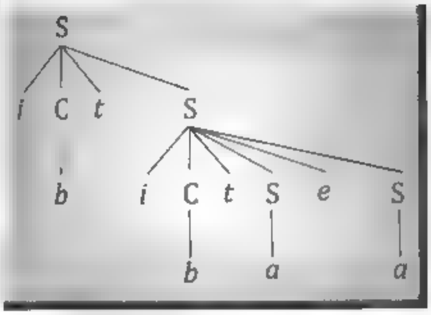
$S \rightarrow iCtS \mid iCtSeS \mid a$

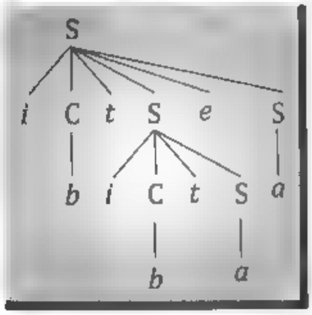
$C \rightarrow b$

Solution

The string *ibtibtaea* can be obtained by applying left most derivation in different ways. The corresponding parse tree for the two left derivations are shown below.

$S \rightarrow iCtS$	$S \rightarrow iCtSeS$
$\rightarrow ibtS$	$\rightarrow ibtSeS$
$\rightarrow ibtiCtSeS$	$\rightarrow ibtiCtSeS$
$\rightarrow ibtibtSeS$	$\rightarrow ibtibtSeS$
$\rightarrow ibtibtaeS$	$\rightarrow ibtibtaeS$
$\rightarrow ibtibtaea$	$\rightarrow ibtibtaea$





Since two parse trees are different for the same string *i b t i b t a e a*, the grammar is ambiguous.

PROBLEM 6

Is the following grammar ambiguous?

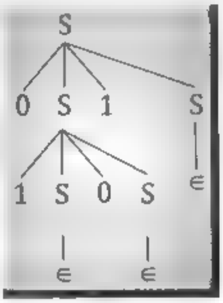
$$S \rightarrow 0S1S \mid 1S0S \mid \epsilon$$

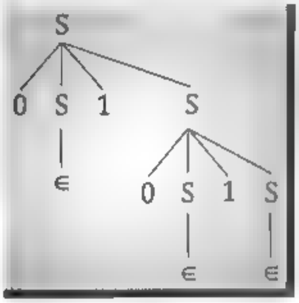
SOLUTION

Consider the two left most derivation for the string 0 1 0 1

$S \rightarrow 0S1S$	$S \rightarrow 0S1S$
$\rightarrow 01S0S1S$	$\rightarrow 01S$
$\rightarrow 010S1S$	$\rightarrow 010S1S$
$\rightarrow 0101S$	$\rightarrow 0101S$
$\rightarrow 0101$	$\rightarrow 0101$

The corresponding parse tree for the two left most derivations are shown below





Since the two parse trees are different the given grammar is ambiguous.

PROBLEM 7

Obtain the unambiguous grammar for the grammar shown

$$E \rightarrow E+E \mid E-E$$
$$E \rightarrow E*E \mid E \mid E$$
$$E \rightarrow (E) \mid I$$
$$I \rightarrow a \mid b \mid c$$

SOLUTION

The grammar can be converted into unambiguous grammar based on the precedence. The identifiers a, b and c have the highest precedence, then the expression within "(" and ")" and then * and/which ever occurs first from left to right and finally + or – whichever occurs first from left to right.

By assuming all these operators are left associative,

$$I \rightarrow a \mid b \mid c$$
$$F \rightarrow (E) \mid I$$
$$T \rightarrow T * F \mid T / F \mid F$$
$$E \rightarrow E + T \mid E - T \mid T$$

so the final grammar which is unambiguous is shown below.

$$E \rightarrow E + T \mid E - T \mid T$$
$$T \rightarrow T * F \mid T / F \mid F$$
$$F \rightarrow (E) \mid I$$
$$I \rightarrow a \mid b \mid c$$

Problem 6

Consider the following grammar

$$R \rightarrow R/R \mid R{\cdot}R \mid R^* \mid (R) \mid a \mid b$$

i) **Construct left sentential form and right sentential form for the string $(a/b)^* abb$**

ii) **Is the following grammar ambiguous, If so construct an equivalent unambiguous grammar.**

SOLUTION

i)

Left sentential form

$R \rightarrow R{\cdot}R$
$\rightarrow R^*{\cdot}R$
$\rightarrow (R)^*{\cdot}R$
$\rightarrow (R / R)^*{\cdot}R$
$\rightarrow (a / R)^*{\cdot}R$
$\rightarrow (a / b)^*{\cdot}R$
$\rightarrow (a / b)^*{\cdot}R.R$
$\rightarrow (a / b)^*{\cdot}a\, R$
$\rightarrow (a / b)^*{\cdot}a\, R{\cdot}R$
$\rightarrow (a / b)^*{\cdot}a\, b\, R$
$\rightarrow (a / b)^*{\cdot}a\, b\, b$

Right sentential form

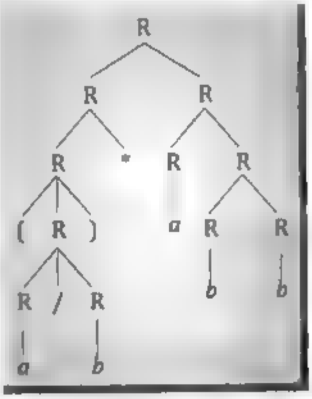
$R \rightarrow R{\cdot}R$
$\rightarrow R{\cdot}b$
$\rightarrow R{\cdot}R\, b$
$\rightarrow R{\cdot}b\, b$
$\rightarrow R{\cdot}R\, b\, b$
$\rightarrow R{\cdot}a\, b\, b$
$\rightarrow R^*{\cdot}\, a\, b\, b$
$\rightarrow (R)^*{\cdot}\, a\, b\, b$
$\rightarrow (R/R)^*{\cdot}a\, b\, b$
$\rightarrow (R/b)^*\, a\, b\, b$
$\rightarrow (a/b)^*\, a\, b\, b$

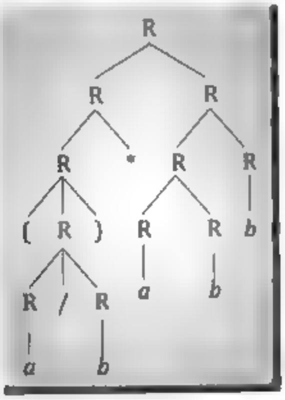
ii) **Checking ambiguity**

The sentence $(a/b)^*\ abb$ can be obtained from left most derivation in two ways as shown below

$R \rightarrow R.R$	$R \rightarrow R.R$
$\rightarrow R^*.R$	$\rightarrow R^*.R$
$\rightarrow (R)^*.R$	$\rightarrow (R)^*.R$
$\rightarrow (R/R)^*.R$	$\rightarrow (R/R)^*.R$
$\rightarrow (a/R)^*.R$	$\rightarrow (a/R)^*.R$
$\rightarrow (a/b)^*.R$	$\rightarrow (a/b)^*.R$
$\rightarrow (a/b)^*.R.R$	$\rightarrow (a/b)^*.R.R$
$\rightarrow (a/b)^*\ aR$	$\rightarrow (a/b)^*\ R.R.R$
$\rightarrow (a/b)^*\ aR.R$	$\rightarrow (a/b)^*\ a\ R\ R$
$\rightarrow (a/b)^*\ ab.R$	$\rightarrow (a/b)^*\ a\ b\ R$
$\rightarrow (a/b)^*\ abb$	$\rightarrow (a/b)^*\ a\ b\ b$

The corresponding parse trees are shown below.





Since there exists two different parse tree for the sentence (a/b)* abb the given grammar is ambiguous.

The unambiguous grammar is given below,

$I \rightarrow a \mid b$

$F \rightarrow I \mid R^*$

$T \rightarrow F \mid (R)$

$R \rightarrow T \mid R.T \mid R/T$

It can be rewritten as

$R \rightarrow R/T \mid R.T \mid T$

$T \rightarrow F \mid (R)$

$F \rightarrow I \mid R^*$

$I \rightarrow a \mid b$

Problem 9

Check whether the following grammar is ambiguous.

$$S \to A\,B \mid C$$
$$A \to a\,A\,b \mid a\,b$$
$$B \to c\,B\,d \mid c\,d$$
$$C \to a\,C\,d \mid a\,D\,d$$
$$D \to b\,D\,c \mid b\,c$$

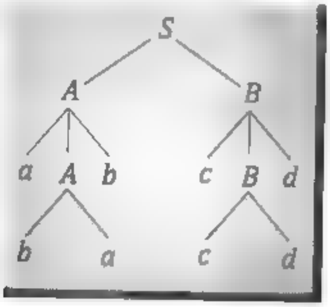
Solution

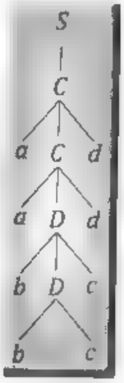
Consider the string *aabbccdd*.

This string can be derived using two different left most derivations as shown below.

$S \to A\,B$	$S \to C$
$\to a\,A\,b\,B$	$\to a\,C\,d$
$\to a\,a\,b\,b\,B$	$\to a\,a\,D\,d\,d$
$\to a\,a\,b\,b\,c\,B\,d$	$\to a\,a\,b\,D\,c\,d\,d$
$\to a\,a\,b\,b\,c\,c\,d\,d$	$\to a\,a\,b\,b\,c\,c\,d\,d$

The corresponding parse trees are





Sınce it has two different derivations the grammar ıs ambiguous.

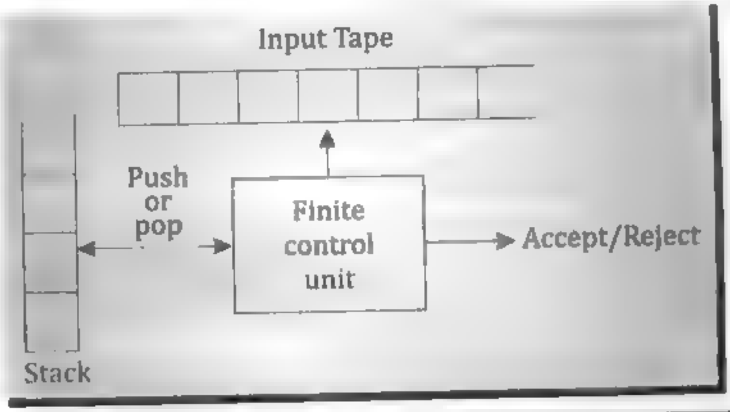
Push Down Automata (PDA)

Just as the regular sets have an equivalent automation - the Finite automata, the context free languages/grammars has their machıne counterpart – the pushdown automation.

A DFA (or NFA) is not powerful enough to recognize many context-free languages, since the finite automation have strıct finıte memories. The automation to recognize the CFL may require additional amount of storage.

A DFA (or NFA) has limitations that they can not count and can not store the input for future reference, so we need a new machine called Push down Automation (PDA) to recognise CFL. PDA is a finite automata with the addition of a stack. A PDA has three components

- an input tape
- a control unit
- a stack with infinite size



Definition : Push Down Automata

A Push Down Automata (PDA) is a seven tuple

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

Where

Q – is a set of finite states

Σ – set of input alphabets

Γ – set of stack alphabets

δ – transitions from $Q \times (\Sigma \cup \in) \times \Gamma$ to finite sub set of $Q \times \Gamma^*$

$q_0 \in Q$ is the start state of M

$Z_0 \in \Gamma$ is the initial symbol on the stock

$F \subseteq Q$ is set of final states.

Transitions

The transition function accepts three parameters namely a state, an input symbol and stack symbol and returns a new state after changing the top of the stack. The transition function has the form.

δ (state, input symbol, stack symbol) = (next state, stack symbol)

Example 1:

The transition $\delta(P, a, Z) = (q, aZ)$ means that the PDA is in current state P and after scanning the input symbol a and with Z on top of the stack, the PDA enters into new state q pushing a on to the stack.

Example [illegible]

The transition $\delta(P, a, Z) = (q, \in)$ means that in state P, on scanning the input symbol a, when Z is on top of the stack, the machine enters into state q and topmost symbol Z is poped from the stack

Example [illegible]

The transition $\delta(P, a, Z) = (q, r)$ means that in the state P, on scanning the input symbol a and when the top of the stack is Z, the machine enters into new state q by replacing the topmost stack symbol Z by r.

Example [illegible]

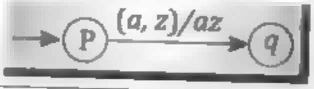
The transition $\delta(P, \in, b) = (P, \in)$ means that in state P, without on input symbol and with b on top of stack the machine remains in same state, and deletes (pop) the topmost stack symbol b.

Graphical representation of PDA

- PDA is represented graphically using the following notations.
- The states of the PDA is represented using circles.
- The start state of the PDA is represented by a circle with an arrow not originating from any node.
- Final states are represented using two concentric circles.
- Transitions are represented by labeled arc. If the transition is $\delta(P, a, Z) = (q, \alpha)$ the arc from state P to q will be labeled as a, Z/α

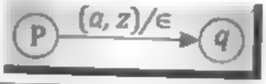
Example 1 :

The graphical representation of $\delta(P, a, Z) = (q, az)$ is shown as



Example 2 :

The graphical representation of $\delta(P, a, Z) = (q, \in)$ can be



Instantaneous Description

The current configuration of PDA at any given instant can be described by an instantaneous description (ID). ID gives the current state of the PDA, the remaining string to be processed, and the entire contents of the stack.

Definition : Instantaneous Description of PDA

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. An ID is defined as triplet.
(q, w, α)

where q – is the current state
w – the remaining string to be processed.
α – is the current contents of the stack

Note: The left most symbol in the string α is on the top of the stack and right most symbol in α is at the bottom of the stack.

Example :

Let ID of a PDA be $(q, aw, Z\alpha)$

It means that

q – is the current state

aw – is the string to be processed

$Z\alpha$ – current contents of the stack with Z on top of the stack.

If there was a transition like $\delta(q, a, z) = (p, \beta)$ then the new configuration obtained is $(p, w, \beta\alpha)$. It is represented as $(q, aw, z\alpha) \vdash (p, w, \beta\alpha)$

i.e., $(q, aw, z\alpha)$ **derives** $(p, w, \beta\alpha)$ **in one move.**

In order to represent more than one move, we use $\vdash^*$ (ie.) $(aw, z\alpha) \vdash^* (p, w, \beta\alpha)$

Language accepted by a PDA

There are two cases where in a string w is accepted by a PDA. They are

- Final state acceptance and
- Empty stack acceptance

Definition:

1. **Final state acceptance:**

   Get the final state from the start state.

   In final state acceptance, the language accepted is the set of all inputs for which some choice of moves causes the push down automata to enter a final state.

   Get the final state from the start state

   Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA The language $L(M)$ accepted by final state is defined as

   $L(M) = \{w \mid (q_0, w, Z_0) \vdash^* (P, \in, \alpha)\}$ where $\alpha \in \Gamma^*, P \in F, w \in \Sigma^*$

   It means, the PDA currently in state q_0, by processing the string w enters into a final state P and the contents of the stack is irrelevant.

2. **Empty stack acceptance**

   Get an empty stack from start state.

   In empty stack acceptance, the language accepted is the set of all imputs for which some sequence of moves causes the push down automata to empty its stack.

   The language $N(M)$ accepted by empty stack is defined as

   $N(M) = \{w \mid (q_0, w, Z_0) \vdash^* (P, \in, \in)\}$ where $w \in \Sigma^*, q_0, P \in Q$

   It means that when the string w is accepted by empty stack the final state is irrelevant, the input string should be completely processed and stack should be empty.

3.13 Construction of PDA

Problem 1

Obtain a PDA to accept the language

$L(M) = \{ w\,c\,w^R \mid w \in (a+b)^* \}$ where w^R is the reverse of w.

Solution

Given that $L(M) = \{w\,c\,w^R\}$

if $w = abb$ then reverse of w is denoted by w^R will be $w^R = bba$.

The Language L will be $w\,c\,w^R$. (i.e.,) *abb c bba*.

Which is a string of palindrome. So we have to construct a PDA which accepts a palindrome consisting of a's and b's with the symbol *c* in middle.

General Procedure

Push all the scanned symbols on to the stack till we find a symbol c. Once we pass the middle string, if the string is palindrome for each input symbol, there should be a corresponding symbol in the stack, pop it. Finally if there is no input and stack is empty, the given string is a palindrome.

Input symbol $\Sigma = \{a, b, c\}$. The first input symbols can be a, b or c.

Let q_0 be the initial state and Z_0 the initial symbol on the stack.

Step 1: For input symbol *a* or *b*

In state q_0 with z_0 on top of the stack, push the input symbols into the stack and remain in q_0.

$\delta(q_0, a, z_0) = (q_0, a\,z_0)$

$\delta(q_0, b, z_0) = (q_0, b\,z_0)$

Once the first input symbol is pushed on to the stack, the top of the stack may be either a or b. Push all the symbols on to the stack and remain in state q_0 till we encounter c. The transitions for this can be

$\delta(q_0, a, a) = (q_0, a\,a)$

$\delta(q_0, a, b) = (q_0, a\,b)$

$\delta(q_0, b, a) = (q_0, b\,a)$

$\delta(q_0, b, b) = (q_0, b\,b)$

Step 2: For input symbol *c*

In state q_0, if the input symbol is c, the top of the stack may be either a or b or z_0 (z_0 when w is a null string), move on to new state q_1 and do not change the contents of the stack. The transitions are

$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_0, c, z_0) = (q_1, z_0)$

Step 3: For input symbols a or b (w^R)

In the state q_1, if the next input symbol is same as the symbol on the top of the stack, delete (pop) the symbol from the top of the stack without changing the state q_1 and repeat the process.

The transitions are

$\delta(q_1, a, a) = (q_1, \in)$

$\delta(q_1, b, b) = (q_1, \in)$

Step 4: No more input symbols

If there is no more input symbols and if the contents of the stack is z0 (i.e., empty), change the state to q2 which is the final state. The transition is

$\delta(q_1, \in, z_0) = (q_2, z_0)$

Step 5

∴ The PDA M which accepts the language wcw^R is given by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Where,

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b, c\}$

$\Gamma = \{a, b, z_0\}$

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{q_2\}$

δ: is shown below

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, b, z_0) = (q_0, b z_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, c, z_0) = (q_1, z_0)$

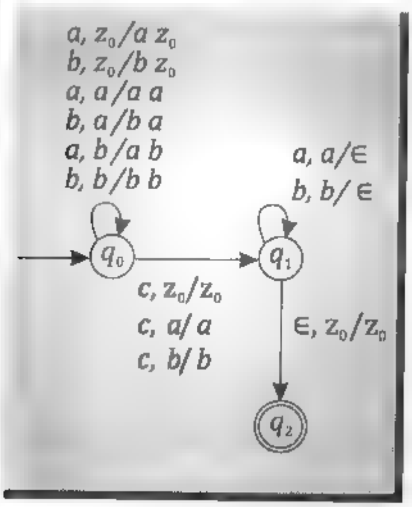
$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_1, a, a) = (q_1, \in)$

$\delta(q_1, b, b) = (q_1, \in)$

$\delta(q_1, \in, z_0) = (q_2, z_0)$



To accept the string

The sequence of moves made by the PDA to check whether the string *aabcbaa* accepted or not.

Initial ID

$(q_0, aabcbaa, z_0) \vdash (q_0, abcbaa, a\, z_0)$

$\vdash (q_0, abcbaa, aa\, z_0)$

$\vdash (q_0, cbaa, baa\, z_0)$

$\vdash (q_1, baa, baa\, z_0)$

$\vdash (q_1, aa, aa\, z_0)$

$\vdash (q_1, a, a\, z_0)$

$\vdash (q_1, \in, z_0)$

$\vdash (q_2, \in, z_0)$

Since q_2 is the final state, and input string is ∈ in the final configuration the strin *aaabcbaa* is accepted by PDA.

To reject the string

The sequence of moves made by PDA for the string aab C bab is shown below

Initial ID

$(q_0, aabcbaa, z_0) \vdash (q_0, abcbab, a\, z_0)$

$\vdash (q_0, bcbab, aa\, z_0)$

$\vdash (q_0, cbab, baa\, z_0)$

$\vdash (q_1, bab, baa\, z_0)$

$\vdash (q_1, ab, aa\, z_0)$

$\vdash (q_1, b, a\, z_0)$

since the transition $\delta(q_1, b, a)$ is not defined, the string *aabcbab* is not a Palindrome.

∴ The string is rejected by the PDA.

Note

The same problem can be converted to accept the language by empty stack only the change is, instead of the final transition

$\delta(q_1, \in, z_0) = (q_2, z_0)$ replace it by the transition.

$\delta(q_1, \in, z_0) = (q_1, \in)$

when the language is accepted by an empty stack, the stack should not contain anything including z_0.

PROBLEM 2

Construct a PDA to accept the language $L = \{a^n b^n \mid n >, 1\}$ by final state.

SOLUTION

iven that L (M) = $\{a^n b^n \mid n \geq, 1\}$ the machine should accept n number of a's followed by n number of b's.

General Procedure

Push all a's into the stack. Once we encounter b's, there should be a corresponding a's on the stack. When the input pointer reaches the end of the string, the stack should be empty. If stack is empty, it indicates that the string scanned has n number of a's followed by n number of b's.

Step 1

Input Symbol $\Sigma = \{a, b\}$

Let q_0 be the initial state and z_0 be the initial symbol on the stack.

In the state q_0, if the input symbol is a, irrespective of the content of the stack. Push a on to the stack. The transitions are,

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, a, a) = (q_0, a a)$

Step 2

In the state q_0, if the next input symbol is b and if the top of the stack is a, then change the state to q_1 and delete one a from the stack.

$\delta(q_0, b, a) = (q_1, \in)$

Step 3

In state q_1, the rest of the symbols to be scanned will be only b's and for each b there should be corresponding a on the stack. If so remain in q1 and delete the corresponding a from the stack.

The transitions are, $\delta(q_1, b, a) = (q_1, \in)$

Step 4

In the state q_1, if the next input symbol to be scanned is $\in$ and if the top of the stack is z_0, change the state to q_2 which is an accepting state. The transitions are

$\delta(q_1, \in, z_0) = (q_2, z_0)$

Step 5

$\therefore$ The PDA M which accepts the language $\{a^n b^n / n > 1\}$

$$\mathbf{M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)}$$

Where,

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, z_0\}$

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{q_2\}$

δ: is shown below

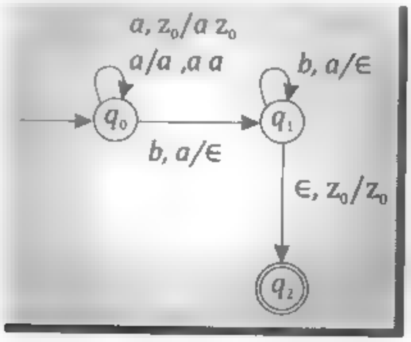
$\delta(q_0, a, z_0) = (q_0, a\, z_0)$

$\delta(q_0, a, a) = (q_0, a\, a)$

$\delta(q_0, b, a) = (q_1, \in)$

$\delta(q_1, b, a) = (q_1, \in)$

$\delta(q_1, \in, z_0) = (q_2, z_0)$



To accept the string

The sequence of moves made by the PDA to check whether the string aaabbb accepted or not.

Initial ID

$$\begin{aligned}(q_0, \text{aaabbb}, z_0) &\vdash (q_0, aabbb, a\, z_0)\\ &\vdash (q_0, abbb, aa\, z_0)\\ &\vdash (q_0, bbb, aaa\, z_0)\\ &\vdash (q_1, bb, aa\, z_0)\\ &\vdash (q_1, b, a\, z_0)\\ &\vdash (q_1, \in, z_0)\\ &\vdash (q_2, \in, z_0)\end{aligned}$$

(Final configuration)

Since q_2 is the final state and the input string is ∈ in the final configuration the strin aaabbb is accepted by PDA.

To reject the string

The sequence of moves made by PDA for the string aabbb is shown below.

Initial ID

$$\begin{aligned}(q_0, aabbb, z_0) &\vdash (q_0, abbb, a\, z_0)\\ &\vdash (q_0, bbb, aa\, z_0)\end{aligned}$$

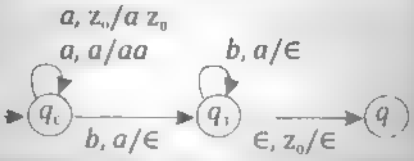
$\vdash (q_1, bb, a\, z_0)$

$\vdash (q_0, b, z_0)$

(Final configuration)

Since the transition $\delta(q_1, b, z_0)$ is not defined, the string *aabbb* is rejected by the PDA.

Note: By changing the final transition from $\delta(q_1, \epsilon, z_0) = (q_2, z_0)$ to $\delta(q_1, \epsilon, z_0)$ (q_2, ϵ) the PDA accepted by an empty stack is obtained. Note that q_2 is not the final state. The corresponding transition diagram accepting by an empty stack is shown below



Problem 3

Obtain a PDA to accept the language L = { $a^n\, b^{2n}$ | $n \geq 1$ } using final state acceptance.

Solution

Given that L = { $a^n\, b^{2n}$ | $n \geq 1$ }, the machine should accept n number of *a*'s followed by 2n number of *b*'s.

General Procedure

For each a in the input, push two *a*'s on to the stack. Once we encounter *b*'s, there should be a corresponding a's on the stack. When the input pointer reaches the end of the string, the stack should be empty. If the stack is empty, it indicates that the string scanned has n number of a's followed by 2n number of *b*'s.

Step 1

Input symbol $\Sigma = \{a, b\}$

Let q_0 be the initial state and z0 be the initial symbol on the stack.

In the state q_0, if the input symbol is a, irrespective of the content of the stack, push two *a*'s on to the stack.

The transitions are

$\delta(q_0, a, z_0) = (q_0, aa\, z_0)$

$\delta(q_0, a, a) = (q_0, aaa)$

Step 2

In the state q_0, if the next input symbol is b and if the top of the stack is a, then chang the state to q_1, and delete one a from the stack.

$\delta(q_0, b, a) = (q_1, \in)$

Step 3

In state q_1, the rest of the symbols to be scanned will be only b's, and for each there should be corresponding a top of the stack. If so remain in q_1 and delete th corresponding a from the stack.

The transitions are $\delta(q, b, a) = (q_1, \in)$

Step 5

In the state q_1, if the next input symbol to be scanned is $\in$ and if the top of the stack is z_0, change the state to q_2 which is an accepting state.

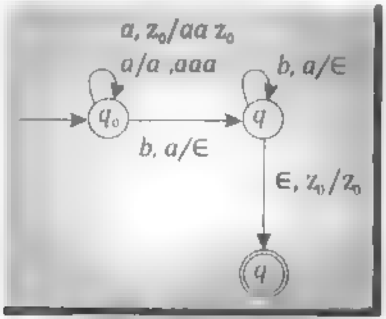
The transitions are $\delta(q, \in, z_0) = (q_2, z_0)$

Step 6

∴ The PDA, M which accepts the language $\{ a^n b^{2n} \mid n \geq 1 \}$

$$\mathbf{M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)}$$

Where,	δ: is shown below
$Q = \{q_0, q_1, q_2\}$	$\delta(q_0, a, z_0) = (q_0, aa\, z_0)$
$\Sigma = \{a, b\}$	$\delta(q_0, a, a) = (q_0, aaa)$
$\Gamma = \{a, z_0\}$	$\delta(q_0, b, a) = (q_1, \in)$
$q_0 = \{q_0\}$	$\delta(q_1, b, a) = (q_1, \in)$
$z_0 = \{z_0\}$	$\delta(q_1, \in, z_0) = (q_2, z_0)$
$F = \{q_2\}$	



Problem 4

Construct a PDA to accept equal number of a's and b's.

(i.e.,) $L = \{w \mid w \in (a + b)^* \text{ and } no_a(w) = no_b(w)\}$

Solution

The language accepted by the machine should consists of strings of a's and b's of any length. Only restriction is number of a's in the string w should be equal to number of b's, but the order is irrelevant.

General procedure

The first scanned input symbol is pushed on to the stack. If the next scanned input symbol is same as on the top of the stack, push the current symbol on to the stack, otherwise pop one symbol from the stack and repeat the process. When end of string is encountered if the stack is empty, the string has equal number of a's and b's.

Step 1

Input symbol $\Sigma = \{ a, b \}$

Let q_0 be the initial state and z0 be the initial stack symbol.

In state q_0 with z_0 on the top of the stack, push the input symbol on to the stack. The transitions are

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, b, z_0) = (q_0, b z_0)$

Step 2

If input symbol is same as the symbol on top of the stack, push the current input symbol to the stack and remain in q0 only, else pop the top element from the stack The transitions are,

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, a, b) = (q_0, \in)$

$\delta(q_0, b, a) = (q_0, \in)$

Step 3

If the next input symbol is $\in$ and top of the stack is z_0 then move to a new accepting state q_1, the transition can be of the form,

$\delta(q_0, \in, z_0) = (q_1, z_0)$

Step 4

$\therefore$ The PDA M which accepts the language $L(M) = \{w \mid w \in (a + b)^* \text{ and } n_a(w) = n_b(w)\}$ is given by $\mathbf{M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)}$

Where,

$Q = \{q_0, q_1\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, b, z_0\}$

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{q_1\}$

δ: is shown below

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, b, z_0) = (q_0, b z_0)$

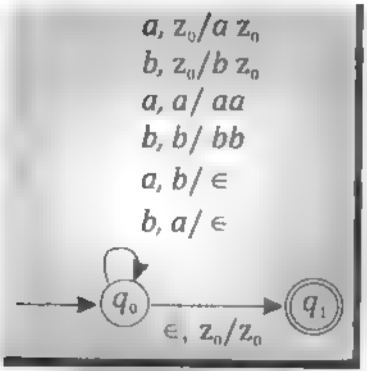
$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, a, b) = (q_0, \in)$

$\delta(q_0, b, a) = (q_0, \in)$

$\delta(q_0, \in, z_0) = (q_1, z_0)$



Problem 5

Construct a PDA to accept a string of balanced parentheses. The parantheses to b considered are (,), [,]

Solution

Some of valid strings are [( ) ( ) ([ ])], ∈, [ ] ([ ])

General procedure

Push each left parentheses on to the stack, if we find right parentheses which matches wit top element of the stack. Pop top of the stack. When the input pointer reaches the end of th string, the stack should be empty.

Step 1

Input symbol $\Sigma = \{\, (,), [,]\,\}$

Let q_0 be the initial state and z_0 be the initial symbol on the stack.

In the state q_0, if the first scanned parenthesis is '(' or '[' push the scanned symbol on to the stack, and change the state to q_1. The transition defined for this can be of the form,

$\delta\,(q_0, (, z_0) = (q_1, (\,z_0)$
$\delta\,(q_0, [, z_0) = (q_1, [\,z_0)$

Step 2

If the scanned input symbol is left parentheses like (or [push it on to the stack. The transitions are

$\delta\,(q_1, (, () = (q_1, (\,()$
$\delta\,(q_1, [, [) = (q_1, [\,[)$
$\delta\,(q_1, (, [) = (q_1, (\,[)$
$\delta\,(q_1, [, () = (q_1, [\,()$

Step 3

If the scanned symbol is right parenthesis like) or] and if there is a matching left parenthesis on the top of the stack, pop the element from the stack. The transition can be defined as

$\delta(q_1,], () = (q_1, \in)$

$\delta(q_1,], [) = (q_1, \in)$

Step 4

When top of the stack is z_0, and there is no more input symbols, change the state to q_2 which is the final state. The transition is $\delta(q_1, \in, z_0) = (q_2, z_0)$

Step 5

∴ The PDA M which accept a string of balanced parentheses is given by

$$\mathbf{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Where,

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{ (,), [,] \}$

$\Gamma = \{ (, [, z_0 \}$

$q_0 = \{ q_0 \}$

$z_0 = \{ z_0 \}$

$F = \{ q_2 \}$

δ: is shown below

$\delta(q_0, (, z_0) = (q_1, (z_0)$

$\delta(q_0, [, z_0) = (q_1, [z_0)$

$\delta(q_1, (, () = (q_1, (()$

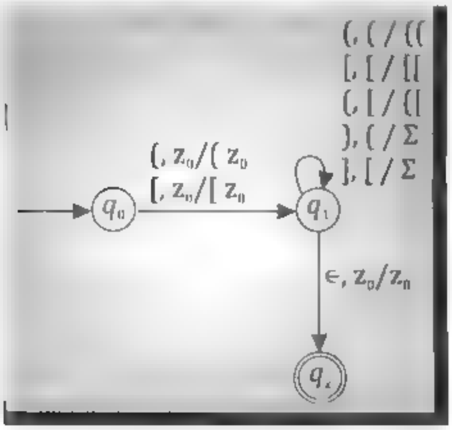
$\delta(q_1, [, [) = (q_1, [[)$

$\delta(q_1, (, [) = (q_1, ([)$

$\delta(q_1,), () = (q_1, \in)$

$\delta(q_1,], [) = (q_1, \in)$

$\delta(q_1, \in, z_0) = (q_2, z_0)$



Problem 6

Obtain a PDA to accept the language L (M) = { $w\, w^R$ | $w \in (a + b)^*$ } where w^R is the reverse of w by final state acceptance.

Solution

Given that L(M) = { $w\, w^R$ }

if $w = abb$ then reverse of w is denoted by w^R will be $w^R = bba$.

The language L will be ww^R = abb bba.

General Procedure

Push all the scanned symbols on to the stack, till we find mid point. Once we pass the middle string, for the reverse string for each input symbol, there should be a corresponding symbol on the stack. Finally if there is no input and stack is empty, the given string is ww^R.

Step 1

Input symbol $\Sigma = \{a, b\}$

Let q_0 be the initial state and z_0 be the initial symbol on the stack.

In the state q_0 with z_0 on the top of the stack, push the input symbols into the stack and remain in q_0

$\delta\,(q_0, a, z_0) = (q_0, a\, z_0)$

$\delta\,(q_0, b, z_0) = (q_0, b\, z_0)$

Now in state q_0, push the input symbols a or b to the stack, irrespective of the stack symbols. The transition are

$\delta\,(q_0, a, a) = (q_0, aa)$

$\delta\,(q_0, a, b) = (q_0, ab)$

$\delta\,(q_0, b, a) = (q_0, ba)$

$\delta\,(q_0, b, b) = (q_0, bb)$

Step 2

Once we reach the midpoint, if the next input symbol is same as the symbol on the top of stack, move to state q_1

$\delta\,(q_0, a, a) = (q_1, \in)$

$\delta\,(q_0, b, b) = (q_1, \in)$

Step 3

In the state q_1 repeat step 2 until we find empty input. The transitions are

$\delta\,(q_1, a, a) = (q_1, \in)$

$\delta\,(q_1, b, b) = (q_1, \in)$

Step 4

Finally in state q_1, if the string is a palindrome, there is no input symbol to be scanned and the stack should be empty. Now change the state to q_2. The transition for this can be,

$\delta(q_1, \in, z_0) = (q_2, z_0)$

Step 5

∴ The PDA, M to accept the language L (M) = $\{ w w^R \mid w \in (a + b)^* \}$ is given by

$$\mathbf{M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)}$$

Where

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

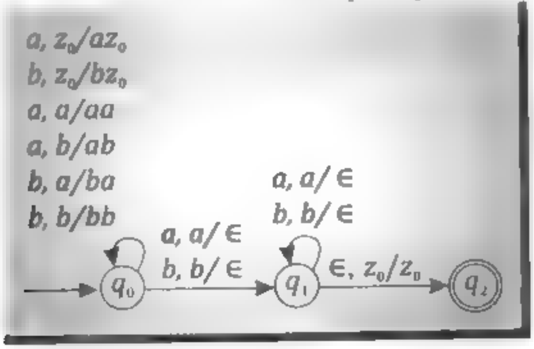
$\Gamma = \{a, b, z_0\}$

$z_0 = \{z_0\}$

$F = \{q_2\}$

δ: is shown below

$\delta(q_0, a, z_0) = (q_0, a z_0)$	$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, b, z_0) = (q_0, b z_0)$	$\delta(q_0, a, a) = (q_1, \in)$
$\delta(q_0, a, a) = (q_0, aa)$	$\delta(q_0, b, b) = (q_1, \in)$
$\delta(q_0, a, b) = (q_0, ab)$	$\delta(q_1, a, a) = (q_1, \in)$
$\delta(q_0, b, a) = (q_0, ba)$	$\delta(q_1, b, b) = (q_1, \in)$
	$\delta(q_1, \in, z_0) = (q_2, z_0)$



3.14 Deterministic Push Down Automata

Let M = $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The PDA is deterministic if the following conditions are satisfied.

1. $\delta(q, a, z)$ has only one element
2. If $\delta(q, \in, z)$ is not empty, then $\delta(q, a, z)$ should be empty.

Problem 1

Is the PDA to accept the language L (M) = { $w c w^R$ | $w \in (a + b)^*$ } deterministic?

Solution

The transitions defined for this machine are, (refer Problem 3.13.1)

$\delta(q_0, a, z_0) = (q_0, a z_0)$	$\delta(q_0, c, z_0) = (q_1, z_0)$
$\delta(q_0, b, z_0) = (q_0, b z_0)$	$\delta(q_0, c, a) = (q_1, a)$
$\delta(q_0, a, a) = (q_0, aa)$	$\delta(q_0, c, b) = (q_1, b)$
$\delta(q_0, b, a) = (q_0, ba)$	$\delta(q_1, a, a) = (q_1, \in)$
$\delta(q_0, a, b) = (q_0, ab)$	$\delta(q_1, b, b) = (q_1, \in)$
$\delta(q_0, b, b) = (q_0, bb)$	$\delta(q_1, \in, z_0) = (q_2, z_0)$

i) The first condition to be deterministic is $\delta(q, a, z)$ should have only one element. In this problem. For each $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$, there is only one element defined.

ii) The second condition states that if $\delta(q, \in, z)$ is not empty, (i.e.) $\delta(q_1, \in, z_0)$ is defined then there should not be any transition from the state q_1 when top of the stack is z_0 which is true.

Since the PDA satisfies both the conditions, the PDA is deterministic.

Problem 2

Is the PDA to accept the language L = {$a^n b^n$ | $n \geq 1$ } deterministic ?

Solution

The transitions defined for this machine are (refer Problem 3.13.2)

$\delta(q_0, a, z_0) = (q_0, aZ)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \in)$

$\delta(q_1, b, a) = (q_1, \in)$

$\delta(q_1, \in, z_0) = (q_2, \in)$

i) The first condition to be deterministic is $\delta(q, a, z)$ should have only one element. In this problem, for each $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$ there is only one element defined.

ii) To satisfy the second condition, consider the transition $\delta(q_1, \in, z_0) = (q_2, \in)$ is defined, the transition $\delta(q_1, a, z_0)$ where $a \in \Sigma$ should not be defined which is true.

Since the PDA satisfies both the conditions. The PDA is deterministic.

PROBLEM 3

Is the PDA to accept the language L = { $a^n b^{2n}$ | $n \geq 1$ } deterministic ?

SOLUTION

The transitions defined for this machine are (refer Problem 3.13.3)

$$\delta(q_0, a, z_0) = (q_0, aaz_0)$$
$$\delta(q_0, a, a) = (q_0, aaa)$$
$$\delta(q_0, b, a) = (q_1, \in)$$
$$\delta(q_1, b, a) = (q_1, \in)$$
$$\delta(q_1, \in, z_0) = (q_2, \in)$$

For each $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$ there exists only one element so the first condition is satisfied.

Since the transition $\delta(q_1, \Sigma, z_0)$ is defined, the transition $\delta(q_1, l, z_0)$ where $a \in \Sigma$ should not be defined which is true.

Since both the conditions are satisfied, the PDA is deterministic.

PROBLEM 4

Is the PDA to accept the language L (M) = {w | $w \in (a+b)^*$ and $n0_a(w) = n0_b(w)$ } is deterministic?

SOLUTION

The transitions defined for this machine are (refer example)

$$\delta(q_0, a, z_0) = (q_0, a\, z_0)$$
$$\delta(q_0, b, z0) = (q_0, b\, z_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, a, b) = (q_0, \in)$$
$$\delta(q_0, b, a) = (q_0, \in)$$
$$\delta(q_0, \in, z_0) = (q_1, z_0)$$

For each $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$ there exists only one element, so the first condition is satisfied.

Since the $\delta(q_0, \Sigma, z_0)$ is defined, the transition $\delta(q_0, a, z_0)$ where $a \in \Sigma$ should not be defind. But there are two transitions.

$$\delta(q_0, a, z_0) = (q_0, a\, z_0)$$
$$\delta(q_0, b, z_0) = (q_0, b\, z_0)$$

defined from q_0 when top of the stack is z_0. Since the second condition is not satisfied, the given PDA is non-deterministic PDA.

Problem 5

Is the PDA to accept the language L = { $w\,w^R$ | $w \in (a + b)^*$ } is deterministic ?

Solution

The transitions defined for this machine are,

$$\begin{aligned}
\delta(q_0, a, z_0) &= (q_0, a\,z_0)\\
\delta(q_0, b, z_0) &= (q_0, b\,z_0)\\
\delta(q_0, a, a) &= (q_0, aa), (q_1, \in)\\
\delta(q_0, a, b) &= (q_0, ab)\\
\delta(q_0, b, a) &= (q_0, ba)\\
\delta(q_0, b, b) &= (q_0, bb), (q_1, \in)\\
\delta(q_1, \in, z_0) &= (q_2, z_0)
\end{aligned}$$

For each $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$ there exists only one element. But there are two transitions having two elements

$$\begin{aligned}
\delta(q_0, a, a) &= (q_0, aa), (q_1, \in)\\
\delta(q_0, b, b) &= (q_0, bb), (q_1, \in)
\end{aligned}$$

So the first condition fails. To be deterministic, both the conditions should be satisfied. So the PDA is non-deterministic PDA.

Short answer questions

1. Formally define a grammar
2. Explain passing
3. Define left most derivation with an example
4. Define right most derivation with an example
5. Write the application of CFG
6. Show that the following grammar is ambiguous.

$$\begin{aligned}
S &\rightarrow a\,S\,b\,S\\
S &\rightarrow b\,S\,a\,S\\
S &\rightarrow \in
\end{aligned}$$

7. Obtain the grammar to generate the language L = { $a^m\, b^m\, c^n$ | $m \geq 1, n \geq 0$ }
8. Define deterministic PDA.

Long answer questions

1. Explain chomsky hiesarchy of generative grammars.
2. Define CFG and the language accepted by a CFG.
3. Define parse tree and subtree with examples.
4. Define PDA, language acceptance of PDA and ID.
5. Explain with an example ambiguous and unambiguous grammar.
6. For the grammar G = (V, T, P, S)

   Where V = {S}, T = {a, b} and P = { $S \rightarrow a\,S\,a \mid b\,S\,b \mid \in$ }

   Construct a leftmost, right most and parse tree.

   a) *aaaaaa*

   b) *abbbba*

   c) *bababbabab*
7. Design a PDA to accept the following language over $\Sigma = \{0, 1\}$

   $L = \{0^n, 1^n \mid n \geq 1\}$

●●●●

UNIT 4

NORMAL FORMS FOR CFG

CHAPTER OUTLINE

INTRODUCTION

a CFG, it is not necessary to use all the symbols in V or all the productions in P, for deriving sentences. So we can reduce the complexity of the grammar, without reducing the generative power of CFG. This is done by simplifying CFG, where in we eliminate useless symbols, ε-productions and unit productions. Normal forms is one way of describing of grammar. In this chapter we explore two types of normal forms - CNF and GNF.

4.1 SUBSTITUTION

n substitution method a non-terminal is replaced by the corresponding symbols on the right-hand side.

Definition : Substitution

Let $G = (V, T, P, S)$ be a context free grammar. Consider the productions.

$$A \rightarrow x_1 B x_2$$
$$B \rightarrow y_1 | y_2 | \ \_\_\_ \ y_n$$

The production $A \rightarrow x_1 B x_2$ can be replaced by $A \rightarrow x_1 y_1 x_2 | x_1 y_2 x_2 | \ \_\_\_ \ x_1 y_n x_2$ and the production $B \rightarrow y_1 | y_2 | \_\_\_\_ y_n$ in P can be deleted.

The resulting productions are added to P_1 and the variables are added to V_1. The language generated by the resulting grammar $G_1 = (V_1, T, P_1, S)$ is same as the language accepted by G. i.e., $L(G_1) = L(G)$

PROBLEM 1

Consider the production $A \rightarrow aBa$, $B \rightarrow ab|b$ simplify the grammar by substitution method.

SOLUTION

In the production, $A \rightarrow a\,B\,a$

B can be replaced by the production $B \rightarrow ab|b$ as shown below

$$A \rightarrow aaba|aba$$

- The resulting grammar, $G = (V, T, P, S)$

Where $V = \{A\}$

$T = \{a, b\}$

$S = A$

$P = \{A \rightarrow aaba|aba\}$

4.2 - Left Recursion

Definition : Left Recursion

A grammar G is said to be left recursive if there is some non-terminal A such that

$A \overset{+}{\Rightarrow} A\alpha$ for $\alpha \in (V \cup T)^*$

i.e., if the first symbol on the right hand side is a non-terminal same as left hand side and if th derivation is obtained from the same non-terminal, we say that the grammar is left recursive

Left recursion should be eliminated from the grammar, because if it is there the grammar will ente into an infinite loop.

4.2.1 - Elimination of left recursion

Left recursion in a grammar G can be eliminated as follows. Consider the A-productions o the form,

$$A \rightarrow A\alpha_1 | A\alpha_2 | A\alpha_3 \_\_\_\_ |A\alpha_n | B_1 | B_2 \_\_\_\_ | B_m$$

Where B_i's do not start with A. Then the A-productions can be replaced by

$$A \rightarrow B_1 A' | B_2 A' | B_3 A' | \_\_\_\_\_ | B_m A'$$
$$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \alpha_3 A' | \_\_\_\_\_ | \alpha_n A' | \in$$

Note α's do not start with A'.

Problem 2

Eliminate left recursion from the following grammar.

$$E \rightarrow E + T | T$$
$$T \rightarrow T * F | F$$
$$F \rightarrow (E) | id$$

Solution

Consider the production $E \rightarrow E \underbrace{+T}_{\alpha}|\underbrace{T}_{\beta}$

it can be replaced by

$$E \rightarrow T E'$$
$$E \rightarrow + T E' | \in$$

Consider the production $T \rightarrow T * F | F$

it can be replaced by

$$T \rightarrow F T'$$
$$T' \rightarrow * F T' | \in$$

Consider the production $F \rightarrow (E)\ id.$ It is not left recursive

The grammar obtained after eliminating left recursion is

$E \rightarrow T E'$ $\quad E' \rightarrow + T E' \mid \in$

$T \rightarrow F T'$ $\quad T' \rightarrow * F T' \mid \in$

$F \rightarrow (E) \mid id$

Problem 3

liminate left recursion from the following grammar $E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid id$

Solution

$\rightarrow E\underbrace{+E}_{\alpha_1} \mid E\underbrace{+E}_{\alpha_1} \mid E\underbrace{+E}_{\alpha_1} \mid E\underbrace{/E}_{\alpha_4} \mid \underbrace{id}_{\beta_1}$

an be replaced by

$E \rightarrow id\, E'$

$E' \rightarrow + E E' \mid -E E' \mid *E E' \mid /E E' \mid \in$

Problem 4

liminate left recursion from the following grammar

$S \rightarrow Ab \mid a$

$A \rightarrow Ab \mid Sa$

Solution

he non terminal S is not having immediate left recursion, but if we substitute for S in A roduction, it becomes left recursive.

e., $A \rightarrow Ab \mid Sa$ can be written as $A \rightarrow Ab \mid Aba \mid aa$

y substituting S using $S \rightarrow Ab|a$.

ow A production is left recursive and can be eliminated as below

$A \rightarrow a\, a\, A'$

$A' \rightarrow bA' \mid baA' \mid \in$

∴ The grammar obtained after eliminating left recursion is

$S \rightarrow Ab \mid a$

$A \rightarrow a\, a\, A'$

$A' \rightarrow b\, A' \mid b\, a\, A' \mid \in.$

4.3 · Simplification of CFG

In a CFG, it may not be necessary to use all the symbols or all the derivations for der sentences. So we simplify the grammar by eliminating those symbols and production wh are not useful for the derivation of sentences, without reducing the generative power context free grammar. Simplification of CFG is done by

- Eliminating useless symbols.
- Eliminating unit productions and
- Eliminating ∈ - productions.

4.4 Elimination of Useless symbols

Any symbol is useful when it appears on the right hand side, in the production rule an generates some terminal string. If no such derivation exists then it is supposed to be useless symbol.

Useless symbols are those variables or terminals that do not appear in the derivation of terminal string from the start symbol.

Definition : Useless Symbols

In a CFG, $G = (V, T, P, S)$, X is useless, if it does not satisfy either of the following conditions.

(a) $X \overset{*}{\Rightarrow} w$, where w is in T^* or (b) $S \overset{*}{\Rightarrow} \alpha X B \overset{*}{\Rightarrow} w$, in $L(G)$

Example :

Consider the following grammar,

$G = (V, T, P, S)$
$V = \{A, B, S, E, F\}$
$T = \{a, b, c, d\}$
$S = S$
$P = \{ S \to AB,$
$A \to a$
$B \to b$
$E \to c$
$F \to d \}$

In this grammar, the variables E and F are useless because they cannot be reached from S.
Terminals c and d are useless because they are not reachable from S.
Productions $E \to c$ and $F \to d$ are useless because they cannot be reached from S.
Even if we remove these unnecessary variables, terminals and productions the language accepted will be the same.

After simplification the grammar, G' is as follows

$G' = (V', T', P', S)$

Where $V' = \{A, B, S\}$ $T' = \{a, b\}$ $S = S$
$P' = \{S \to A B$ $A \to a$ $B \to b\}$

Procedure for eliminating useless symbols

We can convert a grammar to an equivalent one with no useless symbols by first applying Lemma1 and then Lemma2.

Lemma 1

Given a *CFG*, $G = (V, T, P, S)$ with $L(G) = \phi$, we can find an equivalent CFG, $G' = (V', T', P', S)$, such that for each A in V' there is some w in T^* for which $A \overset{*}{\Rightarrow} w$.

Note	L (G) ≠ ϕ, means G should generate at least one string. Lemma1 states that all the variables in V' should derive a terminal symbol

Proof

Each variable A with production $A \rightarrow w$ in P clearly belongs to V'. If $A \rightarrow X_1, X_2 - - - X_n$ is a production, where each X_i is either a terminal or a variable already placed in V', then a terminal string can be added to V'. The set V' can be computed by the following straight forward iterative algorithm.

Note	P' is the set of all production whose symbols are in $V' \cup T$. V' will contain only those variables which can derive the terminal strings.

Algorithm to calculate V'

```
begin
    OLDV = $\phi$ ;
    NEW V = {A|A $\rightarrow$ w for some w in T*};
    while OLDV $\neq$ NEWV do
    begin
      OLDV = NEWV;
      NEWV = OLDV $\cup$ {A|A $\rightarrow \alpha$ for some $\alpha$ in (T $\cup$ OLDV)*}
    end
    V' = NEW V
end
```

Lemma 2

Given a *CFG*, $G = (V, T, P, S)$ we can find an equivalent *CFG*, $G' = (V', T', P', S)$ such that for each X in $(V' \cup T')$ there exist α and β in $(V' \cup T')^*$ for which $S \overset{*}{\Rightarrow} \alpha X \beta$

Lemma 2 states that all the variables in V' and all the terminals in T' should be reachable from S. P' is the set of all productions whose symbols are in $V' \cup T$.

Now by applying lemma 2 on G_1 we can find G' as follows.

Step 1: Place S in V'

Step 2: If A is placed in V' and $A \rightarrow \alpha_1 \mid \alpha_2 \mid --- \mid \alpha_n$, is a production then add all variat of $\alpha_1\ \alpha_2 --- \alpha_n$ to T'.

Step 3: P' is the set of productions of P containing only symbols of $(V' \cup T')$.

Problem 1

Eliminate the useless symbols in the grammar

$$S \rightarrow aA \mid bB$$
$$A \rightarrow aA \mid a$$
$$B \rightarrow bB$$
$$D \rightarrow ab \mid Ea$$
$$C \rightarrow aC \mid d$$

Solution

Step 1: By applying lemma 1, we can obtain a set of variables from which we get on string of terminals as shown.

OLD V	NEW V	Production's
φ	A, D, E	$A \rightarrow a$ $D \rightarrow ab$ $E \rightarrow d$
A, D, E	A, D, E, S	$S \rightarrow a A$ $A \rightarrow a A$ $D \rightarrow E a$
A, D, E, S	A, D, E, S	-

The resulting productions will be

$$S \rightarrow aA$$
$$A \rightarrow aA \mid a$$
$$D \rightarrow Ea \mid ab$$
$$E \rightarrow d$$

Step 2: Applying lemma 2, we obtain the symbols such that each symbol X is reachab.e from the start symbol S, as shown below

P	T	V
-	-	S
$S \rightarrow a A$	a	S, A
$A \rightarrow a A/a$	a	S, A

∴ The resulting grammar, G' after eliminating useless symbols is G' = (V', T', P', S) Where V' = {S, A}

$T' = \{a\}$

$S = S$

$P' = \{S \rightarrow aA$

$A \rightarrow aA \mid a\}$

PROBLEM 2

Eliminate the useless symbols in the following grammar.

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b \mid C$

$E \rightarrow d$

SOLUTION

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	A, B, E	$A \rightarrow a$
		$B \rightarrow b$
		$E \rightarrow d$
A, B, E	A, B, E, S	$S \rightarrow AB$
A, B, E, S	A, B, E, S	-

The resulting productions are

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

$E \rightarrow d$

Step 2: Apply lemma 2

P	T	V
-	-	S
$S \rightarrow AB$	-	S, A, B
$A \rightarrow a$	a	S, A, B
$B \rightarrow b$	a, b	S, A, B

The resulting grammar G' = (V', T', P', S)

Where $V' = \{S, A, B\}$ $T' = \{a, b\}$ $S = S$

$P' = \{S \rightarrow AB$ $A \rightarrow a$ $B \rightarrow b\}$

PROBLEM 3

Eliminate useless symbols in the grammar

$S \rightarrow AB \mid CA$

$A \rightarrow a$

$B \rightarrow BC \mid AB$

$C \rightarrow aB \mid b$

SOLUTION

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	A, C	$A \to a$
		$C \to b$
A, C	A, C, S	$S \to C A$
A, C, S	A, C, S	-

The resulting productions are

$S \to C A$

$C \to b$

$A \to a$

Step 2: Apply lemma 2

P'	T'	V'
-	-	S
$S \to C A$	-	S, C, A
$C \to b$	b	S, C, A
$A \to a$	b, a	S, C, A

$\therefore$ **The resulting grammar $G' = (V', T', P', S)$**

Where $V' = \{S, C, A\}$ $\quad T' = \{b, a\}$ $\quad S = S$

$P' = \{S \to C A \quad C \to b \quad A \to a\}$

PROBLEM 4

Eliminate useless symbols from

$S \to AB \mid a$

$A \to a$

SOLUTION

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	S, A	$S \to a$
		$A \to a$
S, A	S, A	-

The resulting productions are

$S \to a$

$A \to a$

ep 2: Apply lemma 2

P'	T'	V'
-	-	S
$S \to a$	a	S

The resulting grammar $G' = (V', T', P', S)$

Where $V' = \{S\}$ $T' = \{a\}$ $S = S$ $P' = \{S \to a\}$

Problem 5

minate useless symbols from the following productions

$S \to aAa$

$A \to Sb$

$A \to bCC$

$A \to DaA$

$C \to abb$

$C \to DD$

$E \to aC$

$D \to aDa$

Solution

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	C	$C \to abb$
C	C, A, E	$A \to bCC$ $E \to aC$
C, A, E	C, A, E, S	$S \to aAa$
C, A, E, S	C, A, E, S	$A \to Sb$
C, A, E, S	C, A, E, S	-

The resulting productions are

$S \to aAa$

$A \to Sb \mid bCC$

$C \to abb$

$E \to aC$

Step 2: Apply lemma 2

P'	*T'*	*V'*
-	-	*S*
$S \rightarrow aAa$	*a*	*S, A*
$A \rightarrow Sb$	*a, b*	*S, A*
$A \rightarrow bCC$	*a, b*	*S, A, C*
$C \rightarrow abb$	*a, b*	*S, A, C*

The resulting grammar $G' = (V', T', P', S)$

Where $V' = \{S, A, C\}$

$T' = \{a, b\}$

$S = S$

$P' = \{S \rightarrow a\,A\,a$

$A \rightarrow sb \mid bCC$ $\quad C \rightarrow abb\,\}$

PROBLEM 6

Eliminate useless symbols from

$S \rightarrow aA \mid a \mid Bb \mid cC$

$A \rightarrow aB$

$B \rightarrow a \mid Aa$

$C \rightarrow cCD$

$D \rightarrow ddd$

SOLUTION

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	*S, B, D*	$S \rightarrow a$ $B \rightarrow a$ $D \rightarrow ddd$
S, B, D	*S, B, D, A*	$S \rightarrow Bb$ $A \rightarrow aB$
S, B, D, A	*S, B, D, A*	$S \rightarrow aA$ $B \rightarrow Aa$
S, B, D, A	*S, B, D, A*	-

The resulting productions are

$S \rightarrow aA \mid Bb \mid a$

$A \rightarrow aB$

$B \rightarrow Aa \mid a$

$D \rightarrow ddd$

tep 2: Apply lemma 2

P'	T'	V'
-	-	S
$S \to aA$	a	S, A
$S \to Bb$	a, b	S, A, B
$S \to a$	a, b	S, A, B
$A \to aB$	a, b	S, A, B
$B \to Aa$ $B \to a$	a, b	S, A, B

The resulting grammar $G' = (V', T', P', S)$

Where $V' = \{S, A, B\}$ $T' = \{a, b\}$ $S = S$

$P' = \{S \to aA \mid Bb \mid a$ $A \to aB$ $B \to Aa \mid a\}$

PROBLEM 7

Eminate the useless symbol from the following grammar

$S \to aS \mid A \mid C$

$A \to a$

$B \to aa$

$D \to aCb$

SOLUTION

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	A, B	$A \to a$ $B \to aa$
A, B	A, B, S	$S \to A$
A, B, S	A, B, S	$S \to aS$

The resulting productions are

$S \to A \mid aS$

$A \to a$

$B \to aa$

Step 2: Apply lemma 2

P'	T'	V'
-	-	S
$S \to A$	--	S, A
$S \to aS$	a	S, A
$A \to a$	a	S, A

∴ **The resulting grammar $G' = (V', T', P', S)$**

Where $V' = \{S, A\}$ $T' = \{a\}$ $S = S$

$P' = \{S \to A \mid aS$ $A \to a\}$

PROBLEM 8

From the following grammar eliminate useless symbols.

$S \rightarrow A11B \mid 11A$

$S \rightarrow AB \mid 11$

$A \rightarrow 0$

$B \rightarrow BB$

SOLUTION

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	S, A	$S \rightarrow 11$ $A \rightarrow 0$
S, A	S, A	$S \rightarrow 11A$
S, A	S, A	-

The resulting productions are

$S \rightarrow 11A \mid 11$

$A \rightarrow 0$

Step 2: Apply lemma 2

P'	T'	V'
-	-	S
$S \rightarrow 11A$	1	S, A
$S \rightarrow 11$	1	S, A
$A \rightarrow 0$	1, 0	S, A

The resulting grammar $G' = (V', T', P', S)$

Where $V' = \{S, A\}$

$T' = \{1, 0\}$ $S = S$

$P' = \{S \rightarrow 11A \mid 11 \quad A \rightarrow 0\}$

PROBLEM 9

Eliminate the useless symbols from the following grammar.

$S \rightarrow aS|A|G$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow aG$

SOLUTION

Step 1: Apply lemma

OLD V	NEW V	Production's
ϕ	A, B	$A \to a$ $B \to aa$
A, B	A, B, S	$S \to A$
A, B, S	A, B, S	$S \to aS$
A, B, S	A, B, S	-

The resulting productions are

$S \to A \mid aS$

$A \to a$

$B \to aa$

Step 2: Apply lemma 2

P'	T'	V'
-	-	S
$S \to A$	-	S, A
$S \to aS$	a	S, A
$A \to a$	a	S, A

The resulting grammar $G' = (V', T', P', S)$

Where $V' = \{S, A\}$ $T' = \{a\}$ $S = S$

$P' = \{S \to A \mid aS$ $A \to a\}$

PROBLEM 10

Eliminate the useless symbol from the following grammar

$S \to XY$

$S \to 0$

$X \to 1$

SOLUTION

Step 1: Apply lemma 1

OLD V	NEW V	Production's
ϕ	S, X	$S \to 0$ $X \to 1$
S, X	S, X	-

∴ The resulting productions are

$$S \to 0$$
$$X \to 1$$

Step 2: Apply lemma 2

P'	T'	V'
-	-	S
$S \to 0$	0	S

The equivalent grammar $G' = (V', T', P', S)$

$V' = \{S\}$ $T' = \{0\}$ $S = S$ $P' = \{S \to 0\}$

4.5 ELIMINATING ∈ - PRODUCTIONS

4.5.1 ∈ - Productions

Productions of the form $A \to \in$ are called ∈ - productions. ∈ - productions are undesirable in a context - free grammar, unless an empty string is derived from the start symbol. Suppose the language generated from a grammar does not derive any empty string and the grammar consists of ∈ - productions, such ∈ - productions can be removed.

Definition : ∈ - PRODUCTIONS

Let $G = (V, T, P, S)$ be a context - free grammar. A production in P of the form, $A \to \in$ is called an ∈-production or NULL production.

4.5.2 Nullable Variable

If A is a variable and $A \overset{*}{\Rightarrow} \in$ [i.e., if A leads to an empty string in zero, one or more derivations], then A is called Nulluable Variable.

Definition : Nullable Variable

Let $G = (V, T, P, S)$ be a context free grammar. A nullable variable is defined as follows.

(i) If $A \to \in$ is a production in P, then A is a nullable variable.

(ii) If $A \to B_1, B_2 \ldots B_n$ is a production in P_1 and if $B_1, B_2 \ldots B_n$ are nullable variables, then A is also a nullable variable.

(iii) The variables for which there are productions of the form shown in step (i) and step (ii) are nullable variables.

ample :

nsider the grammar

$$S \rightarrow ABCa \mid bD$$
$$A \rightarrow BC \mid b$$
$$B \rightarrow b \mid \in$$
$$C \rightarrow c \mid \in$$
$$D \rightarrow d$$

this grammar, the productions

$$B \rightarrow \in$$
$$C \rightarrow \in$$

re $\in$ - productions and the variables B, C are nullable variables. Since there is a production. $A \rightarrow BC$ nd both B and C are nullable variables, then A is also a nullable variable.

ven though a grammar G has some $\in$ - productions, the language may not derive a language ntaining empty string. In such cases, the $\in$ - productions or NULL productions are not needed nd they can be eliminated.

4.5.3 - Method to eliminate $\in$ - production

4.5.3.1 Theorem

$G = (V, T, P, S)$ is a context free grammar, then there exists a context free grammar, G', having o $\in$ - productions such that $L(G') = L(G) - \{\in\}$

Proof

This theorem can be proved or the grammar G' can be obtained by applying step 1 and tep 2.

Step 1: Construction of a set of nullable variables.

The nullable variables, w can be computed recursively as follows:

(i) $w_i = \{A \mid A \text{ is in } V \text{ and } A \rightarrow \in \text{ is in } P\}$

(ii) $w_{i+1} = w_i \cup \{A \mid A \text{ is in } V \text{ and } A \rightarrow \alpha, \text{ for } \alpha \text{ in } w_i^*\}$

(iii) Repeat step (i) and step (ii) until $w_i = w + 1$

Step 2: Construction of P'.

(i) Any production whose right hand side does not have any nullable variables is included in P'.

(ii) For every production of the form. $A \rightarrow \alpha$ in P $[\alpha \in w_i^*]$ add to P' all the productions that can be obtained from this production by deleting any subset of occurances in α of nullable variables.

Since the nullable variables are eliminated, it is clear that G' does not contain any null productions. Hence proved.

PROBLEM 1

Consider the following grammar $G = (V, T, P, S)$

Where $V = \{S, A, B, D\}$

$T = \{a, b\}$

$S = S$

$P = \{S \rightarrow AB \mid aS$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

$D \rightarrow b\}$

Construct a grammar, G' without nullable productions equivalent to the abov grammar.

SOLUTION

Step 1: Construction of a set of nullable variables.

$A \rightarrow \in$

$B \rightarrow \in$

$\therefore \quad w_1 = \{A, B\}$

$w_2 = \{A, B\} \cup \{S\}$

[Since in the production $S \rightarrow AB$, the nullable variables, w, occurs in the Right hand side]

$= \{A, B, S\}$

$w_2 = \{A, B, S\} \cup \phi$

$= \{A, B, S\}$

i.e., set of nullable variables

$w = \{A, B, S\}$

Step 2: Construction of P'

P' is obtained by deleting the subsets of nullable variables from the productions.

Consider, D $\rightarrow$ b

The right hand side of this production does not have any nullable variable

Consider, $S \rightarrow AB$

By deleting the subset of nullable variables it generates.

$S \rightarrow B$

$S \rightarrow A$

Consider S $\rightarrow$ a S it generates $S \rightarrow a$

The equivalent grammar without $\in$ - productions $G' = (V', T, P', S)$

Where $V' = \{S, A, B, D\}$

$T = \{a, b\}$

$S = S$

$P' = \{S \rightarrow AB|B|A|a\,S|a$

$D \rightarrow b\}$

— Problem 2 —

Let $G = (V, T, P, S)$

where $V = \{S, A, B\}$

$T = \{a, b, c\}$

$P = \{S \rightarrow BAA\,B$

$A \rightarrow aAc \mid cAa \mid \in$

$B \rightarrow AB \mid bB \mid \in\}$

Construct G' without $\in$ - productions.

— Solution —

Step 1: Construction of a set of nullable variables

$w_1 = \{A, B\}$ Since $A \rightarrow \in$

$B \rightarrow \in$

$w_2 = \{A, B\} \cup \{S\}$ Since $S \rightarrow BAAB$

$= \{A, B, S\}$

$w_3 = \{A, B, S\} \cup \emptyset$

$= \{A, B, S\}$

$w = \{A, B, S\}$

Step 2: Construction of P'

Consider $S \rightarrow B\,A\,A\,B$,

by deleting one nullable variable it generates

$S \rightarrow A\,A\,B$

$S \rightarrow B\,A\,B$

$S \rightarrow B\,A\,A$

by deleting two nullable variables

$S \rightarrow A\,B$

$S \rightarrow BA$

$S \rightarrow BB$

$S \rightarrow AA$

by deleting three nullable variables.

$S \rightarrow B$

$S \rightarrow A$

consider $A \rightarrow aAc$ it generates $A \rightarrow ac$

consider $A \rightarrow cAa$ it generates $A \rightarrow ca$

consider $B \rightarrow AB$ it generates $B \rightarrow B$

$B \rightarrow A$

consider $B \rightarrow bB$ it generates $B \rightarrow b$

$\therefore$ **The equivalent grammar without $\in$ - productions is $G' = (V', T, P', S)$**

where $V' = \{S, A, B\}$

$T = \{a, b\}$

$S = S$

$P' = \{S \rightarrow BAAB \mid AAB \mid BAB \mid BAA \mid AB \mid BA \mid BB \mid AA \mid B \mid A$

$A \rightarrow aAc \mid ac \mid cAa \mid ca$

$B \rightarrow AB \mid B \mid A \mid bB \mid b\}$

PROBLEM 3

Consider the productions,

$S \rightarrow AB$

$A \rightarrow aAA \mid \in$

$B \rightarrow bBB \mid \in$

Find a *CFG*, without $\in$ - productions.

SOLUTION

Step 1: Construction of a set of nullable variables.

$w_1 = \{A, B\}$ Since $A \rightarrow \in$

$B \rightarrow \in$

$w_2 = \{A, B\} \cup \{S\}$ Since $S \rightarrow AB$

$= \{A, B, S\}$

$w_3 = \{A, B, S\} \cup \emptyset = \{A, B, S\}$

$\therefore \quad w = \{A, B, S\}$

ep 2: Construction of P'

consider $S \rightarrow AB$ it generates $S \rightarrow A, S \rightarrow B$

consider $A \rightarrow a\,A\,A$ it generates $A \rightarrow a\,A, A \rightarrow a$

consider $B \rightarrow b\,B\,B$ it generates $B \rightarrow b\,B, B \rightarrow b$

The equivalent grammar G' without $\in$ - productions. $G' = (V', T, P', S)$

where $V' = \{S, A, B\}$

$T = \{a, b\}$

$S = S$

$P' = \{S \rightarrow AB|A|B$

$A \rightarrow aAA|aA|a$

$B \rightarrow bBB|bB|b\}$

PROBLEM 4

nd a CFG, **without $\in$ - productions equivalent to the following grammar.**

$S \rightarrow A\,B\,C\,a\,|b\,D$

$A \rightarrow B\,C\,|\,b$

$B \rightarrow b\,|\,\in$

$C \rightarrow c\,|\,\in$

$D \rightarrow d$

SOLUTION

Step 1. Construction of a set of nullable variables.

$w_1 = \{B, C\}$ [Since $B \rightarrow \in, C \rightarrow \in$]

$w_2 = \{B, C\} \cup \{A\}$ Since $A \rightarrow BC$]

$= \{A, B, C\}$

$w_3 = \{A, B, C\} \cup \emptyset$ [S $\rightarrow$ $ABC\,a$ cannot be considered since a is present in the = $\{A, B, C\}$ right hand side]

$w = \{A, B, C\}$

Step 2: Construction of P'

$S \rightarrow A\,B\,C\,a$

it generates $S \rightarrow B\,C\,a$

$S \rightarrow A\,C\,a$

$S \rightarrow A\,B\,a$

$S \rightarrow C\,a$

$S \rightarrow A\,a$

$S \rightarrow B\,a$

$S \rightarrow a$

Consider $S \rightarrow bD$ [no null able variables in the R.H.S.]

Consider $A \rightarrow BC$ it generates $A \rightarrow B,\ A \rightarrow C$

$A \rightarrow b$ **[no null able variables in the R.H.S]**

$B \rightarrow b$

$C \rightarrow c$

$D \rightarrow d$

$\therefore$ **The equivalent grammar** $G' = (V', T, P', S)$

where $V' = \{S, A, B, C, D\}$

$T = \{a, b, c, d\}$

$S = S$

$P' = \{S \rightarrow ABCa \mid BCa \mid ACa \mid ABa \mid Ca \mid Ba \mid Aa \mid a \mid bD$

$A \rightarrow BC \mid B \mid C \mid b$

$B \rightarrow b$

$C \rightarrow c$

$D \rightarrow d\ \}$

PROBLEM 5

Eliminate $\in$ - productions from

$S \rightarrow BAAB$

$A \rightarrow 0A2 \mid 2A0 \mid \in$

$B \rightarrow AB \mid 1B \mid \in$

SOLUTION

Step 1: Construction of a set of null able variables.

$w_1 = \{A, B\}$ [Since $A \rightarrow \in, B \rightarrow \in$]

$w_2 = \{A, B\} \cup \{S\}$ [Since $S \rightarrow BAAB$]

$= \{A, B, S\}$

$w_3 = \{A, B, S\} \cup \emptyset$

$= \{A, B, S\}$

$\therefore\ w = \{A, B, S\}$

Step 2: Construction of P'

consider $S \rightarrow BAAB$

it generates $S \rightarrow BAA$

$S \rightarrow AAB$

$S \rightarrow BAB$

$S \rightarrow BB$

$S \to AA$
$S \to AB$
$S \to BA$
$S \to B$
$S \to A$

consider $A \to 0 A 2$ it generates $A \to 0\,2$
consider $A \to 2 A 0$ it generates $A \to 2\,0$
consider $B \to A B$ it generates $B \to A, B \to B$
consider $B \to 1 B$ it generates $B \to 1$

The equivalent grammar G' = (V', T, P', S)

Where $V' = \{S, A, B\}$
$T = \{0, 1, 2\}$
$S = S$
$P' = \{S \to BAAB \mid BAA \mid AAB \mid BAB \mid BB \mid AA \mid AB \mid BA \mid B \mid A$
$A \to 0A2 \mid 2A0 \mid 02 \mid 20$
$B \to AB \mid 1B \mid A \mid B \mid 1\}$

Problem 6

Find a CFG, without ∈ - productions equivalent to the following grammar

$S \to ABaC$
$A \to BC$
$B \to b \mid \in$
$C \to D \mid \in$
$D \to d\}$

Solution

Step 1: Construction of a set of null able variables

$w_1 = \{B, C\}$ [Since $B \to \in, C \to \in$]
$w_2 = \{B, C\} \cup \{A\}$ [Since $A \to BC$]
$= \{A, B, C\}$
$w_3 = \{A, B, C\} \cup \emptyset = \{A, B, C\}$
$\therefore$ $w = \{A, B, C\}$

Step 2: Construction of P'

Production whose R.H.S with no nullable variables are

$B \to b$
$C \to D$
$D \to b$

consider $S \to ABaC$

it generates $S \to ABa$
$S \to BaC$
$S \to AaC$
$S \to aC$
$S \to Aa$
$S \to Ba$
$S \to a$

consider $A \to BC$ it generates, $A \to B, A \to C$

The equivalent grammar without ∈ - productions G'= (V', T, P', S)

where $V' = \{S, A, B, C, D\}$
$T = \{a, b, d\}$
$S = S$
$P' = \{S \to ABaC \mid ABa \mid BaC \mid AaC \mid aC \mid Aa \mid Ba \mid a$
$A \to BC \mid B \mid C$
$B \to b$
$C \to D$
$D \to d\}$

PROBLEM 7

Consider the following grammar and eliminate all ∈ - productions, without changi the language generated by the grammar.

$S \to AaA$
$A \to Sb \mid bcc \mid \in$
$C \to CC \mid abb$

SOLUTION

Step 1: Construction of a set of nullable variables

$w_1 = \{A\}$ [Since $A \to \in$]
$w_2 = \{A\} \cup \emptyset$
$= \{A\}$
$w = \{A\}$

Step 2: Construction of P'

consider $S \to Aa\,A$ it generates $S \to Aa, S \to aA, S \to a,\ A \to sb|bCC$

Nullable variable does not occur in RHS of the production A

The equivalent grammar G'= (V', T, P', S)

where $V' = \{S, A, C\}$
$T = \{a, b\}$
$S = S$

$P' = \{S \rightarrow AaA|Aa|aA|a$

$A \rightarrow Sb|bCC$

$C \rightarrow CC|abb\}$

PROBLEM 8

Remove the ∈ - production from the following CFG

$S \rightarrow XYX$

$X \rightarrow 0X \mid \in$

$Y \rightarrow 1X \mid \in$

SOLUTION

Step 1: Construction of a set of null able variables.

$w_1 = \{X, Y\}$ [Since $X \rightarrow \in, Y \rightarrow \in$]

$w_2 = \{X, Y\} \cup \{S\}$ [Since $S \rightarrow XYX$]

$= \{X, Y, S\}$

$w_3 = \{X, Y, S\} \cup \emptyset$

$= \{X, Y, S\}$

$w = \{S, X, Y\}$

Step 2: Construction of P'

Consider $S \rightarrow XYX$ it generates $S \rightarrow XX$

$S \rightarrow XY$

$S \rightarrow YX$

$S \rightarrow X$

$S \rightarrow Y$

consider $X \rightarrow 0X$ it generates $X \rightarrow 0$

consider $Y \rightarrow 1X$ it generates $Y \rightarrow 1$

∴ **The equivalent grammar G' = (V', T', P', S)**

where $V' = \{S, X, Y\}$

$T' = \{0, 1\}$

$S = S$

$P' = \{S \rightarrow XYX \mid XX \mid XY \mid YX \mid X \mid Y$

$X \rightarrow 0X \mid 0$

$Y \rightarrow 1X \mid 1\}$

PROBLEM 9

For the CFG given below remove the ∈ - productions.

$$S \rightarrow aSa \mid bSb \mid \in$$

SOLUTION

Step 1: Construction of a set of null able variables

$w_1 = \{S\}$ [Since S → ∈]

$w_2 = \{S\} \cup \emptyset$

$= \{S\}$

$w = \{S\}$

Step 2: Construction of P'

consider $S \rightarrow aSa$ it generates $S \rightarrow aa$

consider, $S \rightarrow bSb$ it generates $S \rightarrow bb$

The equivalent grammar G' = (V', T, P', S) where V' = {S}

$T = \{a, b\}$

$P' = \{S \rightarrow aSa|aa|bSb|bb\}$

PROBLEM 10

Eliminate the ∈ - productions from the CFG given below

$A \rightarrow 0B1 \mid 1B1$

$B \rightarrow 0B \mid 1B \mid \in$

SOLUTION

Step 1: Construction of a set of nullable variables.

$w_1 = \{B\}$ [Since B → ∈]

$w_2 = \{B\} \cup \emptyset$

$= \{B\}$

∴ $w = \{B\}$

Step 2: Construction of P'

consider $A \rightarrow 0B\,1$ it generates $A \rightarrow 01$

consider $A \rightarrow 1\,B\,1$ it generates $A \rightarrow 11$

consider, $B \rightarrow 0$ B it generates $B \rightarrow 0$

consider, $B \rightarrow 1$ B it generates $B \rightarrow 1$

∴ **The equivalent grammar G'= (V', T, P', S)**

where $V' = \{A, B\}$

$T = \{0, 1\}$

$S = A$

$P' = \{A \to 0B1|01|1B1|11$

$B \to 0B|0|1B|1\}$

Problem 11

Construct CFG without ∈ - productions from

$S \to a \mid Ab \mid aba$

$A \to b \mid \in$

$B \to b \mid A$

Solution

Step 1: Construction of a set of nullable variables.

$w_1 = \{A\}$ [Since A → ∈]

$w_2 = \{A\} \cup \{B\}$ [Since B → A]

$= \{A, B\}$

$w_3 = \{A, B\} \cup \emptyset$

$= \{A, B\}$

$w = \{A, B\}$

Step 2 : Construction of P'

Consider, $S \to a$ [no nullable variables in RHS]

consider, $S \to Ab$ it generates $S \to b$

consider, $S \to aba$ [no nullable variables in RHS]

consider, $A \to b$ [no nullable variables in RHS]

consider, $B \to b$ [no nullable variables in RHS]

consider, $B \to A$ [no subsets of nullable variables in RHS]

∴ **the equivalent grammar G'= (V', T, P', S)**

where $V' = \{S, A, B\}$

$T = \{a, b\}$

$S = S$

$P' = \{S \to a \mid Ab \mid b \mid aba$

$A \to b$

$B \to b \mid A\}$

Problem 12

Eliminate ∈ - productions from the following CFG.

$S \to X a$

$X \to a X \mid b X \mid \in$

SOLUTION

Step 1: Construction of set of nullable variables.

$w_1 = \{X\}$ [Since $X \to \in$]

$w_2 = \{X\} \cup \emptyset$

$= \{X\}$

$\therefore w = \{X\}$

Step 2: Construction of P

consider $S \to X a$ it generates $S \to a$

consider $X \to a X$ it generates $X \to a$

consider $X \to a X$ it generates $X \to b$

$\therefore$ **The equivalent grammar** $G' = (V', T, P', S)$

where $V' = \{S, X\}$

$T = \{a, b\}$

$S = S$

$P' = \{S \to X a | a$

$X \to aX \mid a \mid bX \mid b\}$

PROBLEM 13

Eliminate $\in$ - productions from the following CFG.

$S \to X Y$

$X \to Zb$

$Y \to bW$

$Z \to AB$

$W \to Z$

$A \to aA | bA / \in$

$B \to Ba | Bb / \in$

SOLUTION

Step 1: Construction of a set of nullable variables

$w_1 = \{A, B\}$ [Since $A \to \in, B \to \in$]

$w_2 = \{A, B\} \cup \{Z\}$ [Since $Z \to A B$]

$= \{A, B, Z\}$

$w_3 = \{A, B, Z\} \cup \{w\}$ [Since $w \to Z$]

$= \{A, B, Z, w\}$

$w_4 = \{A, B, Z, w\} \cup \emptyset = \{A, B, Z, w\}$

$\therefore w = \{A, B, Z, w\}$

Step 2: Construction of P

consider $S \rightarrow XY$	[No nullable variables in RHS]
consider $X \rightarrow Zb$	it generates $X \rightarrow b$
consider $Y \rightarrow bW$	it generates $Y \rightarrow b$
consider $Z \rightarrow AB$	it generates $Z \rightarrow A, Z \rightarrow B$
consider $W \rightarrow Z$	[no nullable variables in RHS]
consider $A \rightarrow aA$	it generates $A \rightarrow a$
consider $A \rightarrow bA$	it generates $A \rightarrow b$
consider $B \rightarrow Ba$	it generates $B \rightarrow a$
consider $B \rightarrow Bb$	it generates $B \rightarrow b$

The equivalent grammar G'= $\{V', T, P', S\}$

where $V' = \{S, X, Y, Z, w, A, B\}$

$T = \{a, b\}$

$S = S$

$P' = \{S \rightarrow XY$

$X \rightarrow Zb \mid b \mid$

$Y \rightarrow bW \mid b$

$Z \rightarrow AB \mid A \mid B$

$W \rightarrow Z$

$A \rightarrow aA \mid a \mid bA \mid b$

$B \rightarrow Ba \mid a \mid Bb \mid b\ \}$

4.6 Eliminating Unit Productions

Consider the production $A \rightarrow B$. The left hand side of the production and right hand side of the production contains only one variable. Such productions are called unit productions.

Definition : Unit Production

Let $G = (V, T, P, S)$ be a context free grammar. Any production in G of the form $A \rightarrow B$, where both A and B are non terminals is a unit production.

Having unit productions in a grammar is undesirable because one variable is simply replaced by another variable and it adds one more step in the parsing process.

Example :

Consider the productions

$A \rightarrow B$

$B \rightarrow aB \mid b$

In this example $A \rightarrow B$ is a unit production and $B \rightarrow aB|b$ are non unit productions. Since B is generated from A, whatever is generated by B can be generated from A also. So we can remove the unit production $A \rightarrow B$ and the resultant production will be $A \rightarrow aB|b$.

4.6.1 – Procedure to eliminate unit productions

A unit production in grammar G can be eliminates using the following steps.

Step 1: Remove all the productions of the form $A \to A$.

Step 2: Add all non unit productions to P_1

Step 3: For each variable A find all variables B such that $A \overset{*}{\Rightarrow} B$

i.e., In the derivation process from A, if we encounter only one variable in a sententia form say B, obtain all such variables.

Step 4: Obtain a dependency graph for the productions obtained in Step 3.

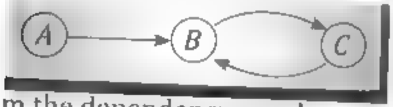
For example, if we have the productions

$A \to B$

$B \to C$

$C \to B$

the dependency graph will be of the form



Step 5: From the dependency graph

(*i*) $A \overset{*}{\Rightarrow} B$

i.e., B can be obtained from A so, all non-unit productions generated from B can also be generated from A.

(*ii*) $A \overset{*}{\Rightarrow} C$

i.e., C can be obtained from A so, all non-unit productions generated from C can also be generated from A.

(*iii*) $B \overset{*}{\Rightarrow} C$

i.e., C can be obtained from B. So, all non-unit productions generated from C can also be generated from B.

(*iv*) $C \overset{*}{\Rightarrow} B$

i.e., B can be obtained from C. So, all non-unit productions generated from B can also be generated from C.

Step 6: Delete all the unit productions from the grammar, G.

Step 7: The resulting grammar, generates the same language as accepted by G.

Problem 1

Eliminate all unit productions from the grammar

$S \rightarrow A\,B$
$A \rightarrow a$
$B \rightarrow C|b$
$C \rightarrow D$
$D \rightarrow E|bC$
$E \rightarrow d|\,Ab$

SOLUTION

Non unit productions

$S \rightarrow A\,B$

$A \rightarrow a$

$B \rightarrow b$

$D \rightarrow b\,C$

$E \rightarrow d|Ab$

Unit productions

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow E$

Dependency graph for unit productions

$$\begin{tikzcd} B \arrow[r] & C \arrow[r] & D \arrow[r] & E \end{tikzcd}$$

From the dependency graph, D $\overset{*}{\Rightarrow}$ E

So all non-unit productions generated from E can also be generated from D.

The non unit productions from E are $E \rightarrow d|Ab$.

this can also be generated from D. i.e., $D \rightarrow d|Ab$.

The resulting D productions are $D \rightarrow b\,C, D \rightarrow d|Ab$

From the dependency grapy, $C \overset{*}{\Rightarrow} E$.

So all the non-unit production from E can also be generated from C.

$C \rightarrow d\,|\,Ab$

From the dependency graph, $C \overset{*}{\Rightarrow} D$.

So all the non-unit productions from D can also be generated from C.

$\therefore \quad C \rightarrow bC\,|\,d\,|\,Ab$

From the dependency graph B $\overset{*}{\Rightarrow}$ C, B $\overset{*}{\Rightarrow}$ D, B $\overset{*}{\Rightarrow}$ E

So all the non-unit productions from C, D and E can also be generated from B.

$\therefore \quad B \rightarrow b\,|\,d\,|\,Ab\,|\,bC$

The final grammar, G', after eliminating unit productions is $G' = (V', T', P', S)$

Where $V' = \{S, A, B, C, D, E\}$

$T' = \{a, b, d\}$

$S = S$

$$\begin{aligned} P = \{ & S \to A\,B \\ & A \to a \\ & B \to b \mid d \mid Ab \mid bC \\ & C \to bC \mid d \mid Ab \\ & D \to bc \mid d \mid Ab \\ & E \to d \mid Ab \,\} \end{aligned}$$

Problem 2

Eliminate all unit productions from the grammar.

$$\begin{aligned} S &\to A\,B \\ A &\to a \\ B &\to C \\ B &\to b \\ C &\to D \\ D &\to E \\ E &\to a \end{aligned}$$

Solution

Non unit productions

$S \to A\,B$
$A \to a$
$B \to b$
$E \to a$

Unit productions

$B \to C$
$C \to D$
$D \to E$

Dependency graph

$$\begin{tikzcd} B \arrow[r] & C \arrow[r] & D \arrow[r] & E \end{tikzcd}$$

From the dependency graph

$D \overset{*}{\Rightarrow} E \qquad \therefore \quad D \to a$

$C \overset{*}{\Rightarrow} E, C \overset{*}{\Rightarrow} D \qquad \therefore \quad C \to a$

$B \overset{*}{\Rightarrow} E, B \overset{*}{\Rightarrow} D, B \overset{*}{\Rightarrow} C \qquad \therefore \quad B \to a$

The simplified grammar $G' = (V', T', P', S)$

Where

$$\begin{aligned} V' &= \{S, A, B, C, D, E\} \\ T' &= \{a, b\} \\ S &= S \\ P' &= \{\, S \to AB \\ & \quad A \to a \\ & \quad B \to a \mid b \end{aligned}$$

$C \to a$
$D \to a$
$E \to a \}$

PROBLEM 3

Eliminate all unit productions from the grammar

$S \to AB$
$A \to D$
$D \to a$
$B \to F$
$F \to b$

SOLUTION

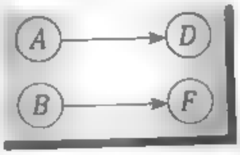
Non unit productions

$S \to AB$
$D \to a$
$F \to b$

Unit Productions

$A \to D$
$B \to F$

Dependency Graph



From the dependency graph

$A \overset{*}{\Rightarrow} D \quad \therefore \quad A \to a$
$B \overset{*}{\Rightarrow} F \quad \therefore \quad B \to b$

The simplified grammar without unit productions is $G' = (V', T', P', S)$
where $V' = \{S, A, B, D, F\}$

$T' = \{a, b\}$
$S = S$
$P' = \{ S \to AB$
$D \to a$
$F \to b$
$A \to a$
$B \to b \}$

PROBLEM 6

Eliminate unit productions from,

$S \rightarrow Aa \mid B \mid Ca$

$B \rightarrow aB \mid b$

$C \rightarrow Db \mid D$

$D \rightarrow E \mid d$

$E \rightarrow ab$

SOLUTION

Non unit productions

$S \rightarrow Aa \mid Ca$

$B \rightarrow aB \mid b$

$C \rightarrow Db$

$D \rightarrow d$

$E \rightarrow ab$

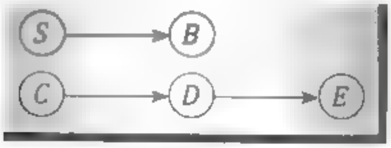
Unit productions

$S \rightarrow B$

$C \rightarrow D$

$D \rightarrow E$

Dependency graph



From the dependency graph

$S \overset{*}{\Rightarrow} B \quad \therefore \quad S \rightarrow aB \mid b$

$D \overset{*}{\Rightarrow} E \quad \therefore \quad D \rightarrow ab \mid d$

$C \overset{*}{\Rightarrow} E, C \overset{*}{\Rightarrow} D \quad \therefore \quad C \rightarrow Db \mid ad \quad d$

The final grammar without unit productions is $G' = (V', T', P', S)$

where $V' = \{S, B, C, D, E, A\}$

$T' = \{a, b, d\}$

$S = S$

$P' = \{ S \rightarrow Aa \mid Ca \mid aB \mid b$

$B \rightarrow aB \mid b$

$C \rightarrow Db \mid ab \mid d$

$D \rightarrow d \mid ab$

$E \rightarrow ab \}$

Problem 5

minate unit productions from the grammar

$S \rightarrow A0 \mid B$

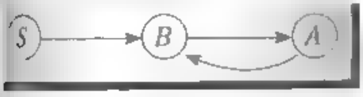
$B \rightarrow A \mid 11$

$A \rightarrow 0 \mid 12 \mid B$

Solution

on-unit productions	Unit Productions
$S \rightarrow A0$	$S \rightarrow B$
$B \rightarrow 11$	$B \rightarrow A$
$A \rightarrow 0 \mid 12$	$A \rightarrow B$

ependency graph



From the dependency graph

$B \overset{*}{\Rightarrow} A \quad \therefore \quad B \rightarrow 11 \mid 0 \mid 12$

$A \overset{*}{\Rightarrow} B \quad \therefore \quad SA \rightarrow 0 \mid 12 \mid 11$

$S \overset{*}{\Rightarrow} B, S \overset{*}{\Rightarrow} A \quad \therefore \quad S \rightarrow A0 \mid 11 \mid 0 \mid 12$

he resultant grammar $G' = (V', T', P', S)$

here $V' = \{S, A, B\}$

$T' = \{0, 1, 2\}$

$S = S$

$P' = \{ S \rightarrow A0 \mid 11 \mid 0 \mid 12$

$B \rightarrow 11 \mid 0 \mid 12$

$A \rightarrow 0 \mid 12 \mid 11 \}$

Problem 6

liminate unit productions from the grammar

$S \rightarrow A \mid bb$

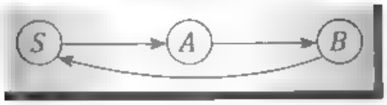
$A \rightarrow B \mid b$

$B \rightarrow S \mid a$

Solution

on-unit productions	Unit Productions
$S \rightarrow bb$	$S \rightarrow A$
$A \rightarrow b$	$A \rightarrow B$
$B \rightarrow a$	$B \rightarrow S$

Dependency Graph



From dependency graph

$A \overset{*}{\Rightarrow} B, A \overset{*}{\Rightarrow} S$	$\therefore$	$A \rightarrow b \mid a \mid bb$
$S \overset{*}{\Rightarrow} B, S \overset{*}{\Rightarrow} A$	$\therefore$	$S \rightarrow b \mid a \mid bb$
$B \overset{*}{\Rightarrow} S, B \overset{*}{\Rightarrow} A$	$\therefore$	$B \rightarrow a \mid b \mid bb$

The resultant grammar $\boldsymbol{G' = (V', T', P', S)}$

Where $V' = \{S, A, B\}$

$T' = \{a, b\}$

$S = S$

$P' = \{ S \rightarrow b \mid a \mid bb$

$A \rightarrow b \mid a \mid bb$

$B \rightarrow b \mid a \mid bb \}$

PROBLEM 7

Remove unit production from

$\boldsymbol{S \rightarrow 0A \mid 1B \mid C}$

$\boldsymbol{A \rightarrow 0S \mid 00}$

$\boldsymbol{B \rightarrow 1 \mid A}$

$\boldsymbol{C \rightarrow 01}$

SOLUTION

Non unit productions

$S \rightarrow 0A \mid 1B$

$A \rightarrow 0S \mid 00$

$B \rightarrow 1$

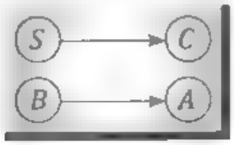
$C \rightarrow 01$

Unit Productions

$S \rightarrow C$

$B \rightarrow A$

Dependency Graph



From dependency graph

$S \overset{*}{\Rightarrow} C$	$\therefore$	$S \rightarrow 0A \mid 1B \mid 01$
$B \overset{*}{\Rightarrow} A$	$\therefore$	$B \rightarrow 1 \mid 0S \mid 00$

The resultant grammar $\boldsymbol{G' = (V', T', P', S)}$

Where $V' = \{S, A, B, C\}$

$T' = \{0, 1\}$

$S = S$

$P' = \{ S \rightarrow 0A \mid 1B \mid 01$

$A \to 0S \mid 00$

$B \to 1 \mid 0S \mid 00$

$C \to 01\}$

Problem 8

nnate unit productions from,

$S \to A \mid 0C1$

$A \to B \mid 01 \mid 10$

$C \to CD$

Solution

n-unit productions

$S \to 0C1$

$A \to 01 \mid 10$

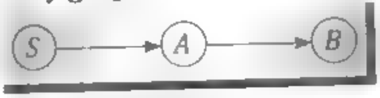
$C \to CD$

Unit productions

$S \to A$

$A \to B$

pendency graph



om dependency graph

$A \overset{*}{\Rightarrow} B \qquad \therefore\ A \to 01 \mid 10$

$S \overset{*}{\Rightarrow} B, S \overset{*}{\Rightarrow} A \qquad \therefore\ S \to 0C1 \mid 01 \mid 10$

e resultant grammar $G' = (V', T', P', S)$

here $V' = \{S, A, C, D\}$

$T' = \{0, 1\}$

$S = S$

$P' = \{ S \to 0C1 \mid 01 \mid 10$

$A \to 01 \mid 10$

$C \to CD \}$

Problem 9

implify the given CFG

$S \to a \mid aA \mid B \mid C$

$A \to aB \mid \wedge$

$B \to aA$

$C \to cCD$

$D \to ddd$

SOLUTION

There is no ∈ - production. Unit productions are there, ∴ so eliminate.

Non unit production	**Unit Productions**
$S \rightarrow a \mid a A$	$S \rightarrow B$
$A \rightarrow a B \mid \wedge$	$S \rightarrow C$
$B \rightarrow a A$	
$C \rightarrow c C D$	
$D \rightarrow ddd$	

Dependency Graph

$$\begin{tikzcd} S \arrow[r] & B \end{tikzcd}$$

From dependency graph

$S \overset{*}{\Rightarrow} B$

∴ $S \rightarrow a A$

$$\begin{tikzcd} S \arrow[r] & C \end{tikzcd}$$

From dependency graph

$S \overset{*}{\Rightarrow} C$

∴ $S \rightarrow c C D$

The resultant productions after eliminating unit productions is

$$\begin{aligned} P' = \{&S \rightarrow a \mid aA \mid cCD \\ &A \rightarrow a B \mid \wedge \\ &B \rightarrow a A \\ &C \rightarrow c C D \\ &D \rightarrow ddd\ \} \end{aligned}$$

Elimination of useless symbols

OLD V	NEW V	Production's
ϕ	S, A, D	$S \rightarrow a$ $A \rightarrow \wedge$ $D \rightarrow ddd$
S, A, D	S, A, D, B	$S \rightarrow a A$ $B \rightarrow a A$
S, A, D, B	S, A, D, B	$A \rightarrow a B$
S, A, D, B	S, A, D, B	-

The resulting productions are

$$\begin{aligned} \{&S \rightarrow a \mid aA \\ &A \rightarrow \wedge \mid aB \\ &B \rightarrow aA \\ &D \rightarrow ddd\} \end{aligned}$$

P'	T'	V'
-	-	S
$S \to a$ $S \to aA$	a	S, A
$A \to \wedge$ $A \to aB$	$a, \wedge$	S, A, B
$B \to aA$	$a, \wedge$	S, A, B

∴ The simplified grammar $G_1 = (V_1, T_1, P_1, S)$

where $V_1 = \{S, A, B\}$

$T_1 = \{a, \wedge\}$

$S = S$

$P_1 = \{ S \to a | aA$

$A \to \wedge | aB$

$B \to aA \}$

Problem 10

Simplify the grammar

$S \to AACD$

$A \to aAb | \wedge$

$C \to aC | a$

$D \to aDa | bDb | \wedge$

Solution

There is no ∈ - production and unit productions.

Elimination of useless symbols

OLD V	NEW V	Production's
ϕ	A, C, D	$A \to \wedge$ $C \to a$ $D \to \wedge$
A, C, D	S, A, C, D	$S \to AACD$ $A \to aAb$ $C \to aC$ $D \to aDa$ $D \to bDb$
S, A, C, D	S, A, C, D	-

The resulting productions are

$S \rightarrow AACD$

$A \rightarrow a\,A\,b|$^

$C \rightarrow a\,C|aD \rightarrow a$

$D\;\; a|b\,D\,b|$^

P'	T'	V'
-	-	S
$S \rightarrow A\,A\,C\,D$	-	S, A, C, D
$A \rightarrow a\,A\,b|$ ^ $C \rightarrow a\,C|a$ $D \rightarrow a\,D\,a|b\,D\,b\,|$ ^	$a, b,$ ^	S, A, C, D

Since there is no useless symbols, the simplified grammar is the same as the given grammar

4.7 NORMAL FORMS FOR CONTEXT - FREE GRAMMARS

When the productions in context free grammars are made to satisfy certain restrictions, then CFG is said to be in normal form.

Some of the normal forms are

- Chomsky Normal form and
- Greibach Normal form

4.8 CHOMSKY NORMAL FORM [CNF]

CNF puts restrictions on number of symbols on the right hand side of a production. That is in CNF the right hand side of a production cannot contain more than two symbols. It can be a single terminal or two nonterminals.

Definition : CNF

Let $G = (V, T, P, S)$ be a context free grammar The grammar G is said to be in CNF if all the productions are of the form.

$A \rightarrow BC$ (or) $A \rightarrow a$

Where A, B and C are non terminals and a is a terminal.

Note The grammar should not have, unit productions, $\in$ - productions and useless symbols.

Example

The grammar

$S \rightarrow AS|a$

$A \rightarrow SA|b$ is in CNF.

The grammar

$S \rightarrow AS|AAS$

$A \rightarrow SA|aa$

is not in CNF beacuse *$S \rightarrow AAS$ contains three nonterminals and $A \rightarrow aa$* contains two terminals which is not allowed in CNF.

4.8.1 Reduction of CFG to CNF

CFG can be converted to CNF using the following theorem.

4.8.1.1 Theorem

For every CFL, without $\in$ generated by a grammar G, there is an equivalent grammar is in CNF such that $L(G) = L(G')$

Proof

Let $G = (V, T, P, S)$ be the CFG generating a language. We construct $G' = (V', T', P', S)$ which is in CNF as follows.

Step 1: Eliminate $\in$ - productions, unit productions and useless symbols from the grammar G.

> **Note** Every production of such grammar is either of the form A → a, which is already a CNF, or it has a body of length two or more.

Step 2: Arrange in such a way that all bodies of length two or more consist only of variables.

For every terminal, 'a' that appears in a body of length two or more, create a new variable say A with a production $A \rightarrow a$. Now use A in place of a. So that every productions has a body i.e., either a single terminal or atleast two variables and no terminals.

Step 3: Break bodies of length three or more into a cascade of productions, each with a body consisting of two variables.

Break those productions $A \rightarrow B_1 B_2 \_\_\_ B_K$, for $K \geq 3$ into a group of productions with two variables in each body.

i.e., introduce K - 2 new variables. $C_1 C_2 \_\_\_ C_{K-2}$.

The original production is replaced, by the K - 1 productions

$A \rightarrow B_1 C_1$

$C_1 \rightarrow B_2 C_2 \_\_\_ C_{K-3}$

$\rightarrow B_{K-2}\, C_{K-2}$

$C_{K-2} \rightarrow B_{K-1}\, B_K$

Thus we get G_1 in CNF. Hence proved.

Problem 1

Consider the grammar $G = (\{S, A, B\}, \{a, b\}, P, S)$

where $P = \{S \rightarrow b\, A \mid a\, B$

$A \rightarrow b\, A\, A \mid a\, S \mid a$

$B \rightarrow aBB \mid bS \mid b$

Find an equivalent grammar in CNF.

Solution

Step 1: No useless symbols, $\in$ - productions or unit productions.

$A \rightarrow a$

$B \rightarrow b$

These productions are already in CNF.

Step 2: Replace terminals by variables

$S \rightarrow b\, A$	is replaced by	$S \rightarrow C_b\, A,$	$C_b \rightarrow b$
$S \rightarrow a\, B$	is replaced by	$S \rightarrow C_a\, B,$	$C_a \rightarrow a$
$A \rightarrow b\, A\, A$	is replaced by	$A \rightarrow C_b\, A\, A,$	$C_b \rightarrow b$
$A \rightarrow a\, S$	is replaced by	$A \rightarrow C_a\, S,$	$C_a \rightarrow a$
$B \rightarrow a\, B\, B$	is replaced by	$B \rightarrow C_a\, B\, B,$	$C_a \rightarrow a$
$B \rightarrow b\, S$	is replaced by	$B \rightarrow C_b\, S,$	$C_b \rightarrow b$

Step 3: Restrict the number of variables on right hand side to two.

$A \rightarrow C_b\, A\, A$	is replaced by	$A \rightarrow C_b\, D_1,$	$D_1 \rightarrow A\, A$
$B \rightarrow C_a\, B\, B$	is replaced by	$B \rightarrow C_a\, D_2,$	$D_2 \rightarrow B\, B$

The final grammar in CNF is obtained by combining all the productions in Step1 Step 2 and Step 3.

The grammar in CNF $G = (V', T', P', S)$

Where $V' = \{S, A, B, C_a, C_b, D, D_2\}$ $\quad T' = \{a, b\}$ $\quad S = S$

$P' = \{S \rightarrow C_b\, A \mid C_a\, B$

$A \rightarrow C_a\, S \mid C_b\, D_1 \mid a$

$B \rightarrow C_b\, S \mid C_a\, D_2 \mid b$

$D_1 \rightarrow A\, A$

$D_2 \rightarrow B\, B$

$C_a \rightarrow a$

$C_b \rightarrow b\, \}$

Problem 2

Reduce the following grammar to CNF

$S \rightarrow ABa$
$A \rightarrow aab$
$B \rightarrow Ac$

Solution

Step 1: No unit production, ∈ - productions and useless symbols.

Step 2: Replace terminals by variables.

$S \rightarrow ABa$, is replaced by $S \rightarrow ABC_a$, $C_a \rightarrow a$
$A \rightarrow aab$, is replaced by $A \rightarrow C_aC_aC_b, C_a \rightarrow a$, $C_b \rightarrow b$
$B \rightarrow Ac$, is replaced by $B \rightarrow AC_c$, $C_c \rightarrow c$

Step 3: Restrict the number of variables on the RHS to two

$S \rightarrow ABC_a$ is replaced by $S \rightarrow AD_1$, $D_1 \rightarrow BC_a$
$A \rightarrow C_aC_aC_b$ is replaced by $A \rightarrow C_aD_2$, $D_2 \rightarrow C_aC_b$

The grammar in CNF $G' = (V', T', P', S)$

where $V' = \{S, A, B, D_1, D_2, C_a, C_b, C_c\}$ $T = \{a, b, c\}$ $S = S$

$P' = \{S \rightarrow AD_1$
$A \rightarrow C_aD_2$
$B \rightarrow AC_c$
$D_1 \rightarrow BC_a$
$D_2 \rightarrow C_aC_b$
$C_a \rightarrow a$
$C_b \rightarrow b$
$C_c \rightarrow c\}$

Problem 3

Reduce the grammar into CNF

$S \rightarrow aXX$
$X \rightarrow aS|bS|a$

Solution

Step 1: No useless symbols, ∈ - productions and unit productions $X \rightarrow a$, already in CNF.

Step 2: Replace terminals by variables.

$S \rightarrow aXX$, is replaced by $S \rightarrow C_aXX$, $C_a \rightarrow a$
$X \rightarrow aS$, is replaced by $X \rightarrow C_aS$, $C_a \rightarrow a$

$X \to bS,$ is replaced by $X \to C_b S,$ $C_b \to b$

Step 3: Restrict the number of variables on the RHS to two.

$S \to C_a XX,$ is replaced by $S \to C_a D_1,$ $D_1 \to XX$

- The grammar in CNF $G' = (V', T', P', S)$

where $V' = \{S, X, D_1, C_a, C_b\}$ $T' = \{a, b\}$ $S = S$

$P' = \{ S \to C_a D_1$

$X \to C_a S \mid C_b S$

$D_1 \to XX$

$C_a \to a$

$C_b \to b\}$

Problem 4

Reduce the following grammar into CNF

$S \to 0A \mid 1B$

$A \to 0AA \mid 1S \mid 1$

$B \to 1BB \mid 0S \mid 0$

Solution

Step 1: No useless symbols, ∈ - productions and unit productions

$A \to 1, B \to 0,$ already in CNF

Step 2: Replace terminals by variables

$S \to 0A,$	is replaced by	$S \to C_0 A,$	$C_0 \to 0$
$S \to 1B,$	is replaced by	$S \to C_1 B,$	$C_1 \to 1$
$A \to 0AA$	is replaced by	$A \to C_0 AA,$	$C_0 \to 0$
$A \to 1S,$	is replaced by	$A \to C_1 S,$	$C_1 \to 1$
$B \to 1BB,$	is replaced by	$B \to C_1 BB,$	$C_1 \to 1$
$B \to 0S,$	is replaced by	$B \to C_0 S,$	$C_0 \to 0$

Step 3: Restrict the number of variables on the RHS by two

$A \to C_0 AA,$	is replaced by	$A \to C_0 D_1,$	$D_1 \to AA$
$B \to C_1 BB,$	is replaced by	$B \to C_1 D_2,$	$D_2 \to BB$

- The equivalent grammar in CNF $G = (V, T, P, S)$

where $V = \{S, A, B, D_1, D_2, C_0, C_1\}$ $T = \{0, 1\}$ $S = S$

$P' = \{S \to C_0 A \mid C_1 B$

$A \to C_1 S \mid C_0 D_1 \mid 1$

$B \to C_0 S \mid C_1 D_2 \mid 0$

$D_1 \rightarrow A\,A$

$D_2 \rightarrow B\,B$

$C_0 \rightarrow 0$

$C_1 \rightarrow 1\}$

PROBLEM 5

Reduce the following grammar *G* to CNF. *G* is

$S \rightarrow a\,A\,D$

$A \rightarrow a\,B \mid b\,A\,B$

$B \rightarrow b$

$D \rightarrow d$

SOLUTION

Step 1: No useless symbols, ∈ - productions and unit productions.

$B \rightarrow b$

$D \rightarrow d$, already in CNF.

Step 2: Replace terminals by variables

$S \rightarrow a\,A\,D$, is replaced by $S \rightarrow C_a\,A\,D,\ \ C_a \rightarrow a$

$A \rightarrow a\,B$, is replaced by $A \rightarrow C_a\,B,\ \ C_a \rightarrow a$

$A \rightarrow b\,A\,B$, is replaced by $A \rightarrow C_b\,A\,B,\ \ C_b \rightarrow b$

Step 3: Restrict the number of variables on the RHS by two

$S \rightarrow C_a\,A\,D$, is replaced by $S \rightarrow C_a\,D_1,\ \ D_1 \rightarrow A\,D$

$A \rightarrow C_b\,A\,B$, is replaced by $A \rightarrow C_b\,D_2,\ \ D_2 \rightarrow A\,B$

- The equivalent grammar in CNF $G' = (V, T, P, S)$

Where $V' = \{S, A, B, D, D_1, D_2, C_a, C_b\}$

$T = \{a, b, d\}$ $\quad S = S \quad P' = \{S \rightarrow C_a\,D_1$

$A \rightarrow C_a\,B \mid C_b\,D_2$

$D_1 \rightarrow A\,D$

$D_2 \rightarrow A\,B$

$B \rightarrow b$

$D \rightarrow d$

$C_a \rightarrow a$

$C_b \rightarrow b\}$

Problem 6

Find a grammar in CNF equivalent to

$S \rightarrow a\,A\,b\,B$

$A \rightarrow a\,A \mid a$

$B \rightarrow b\,B \mid b$

Solution

Step 1: No useless symbols, $\in$ - productions and unit productions.

$A \rightarrow a$

$B \rightarrow b$, already in CNF

Step 2: Replace terminals by variables

$S \rightarrow a\,A\,b\,B$ is replaced by

$S \rightarrow C_a\,A\,C_b\,B$

$C_a \rightarrow a$

$C_b \rightarrow b$

$A \rightarrow a\,A$, is replaced by $A \rightarrow C_a\,A$, $C_a \rightarrow a$

$B \rightarrow b\,B$, is replaced by $B \rightarrow C_b\,B$, $C_b \rightarrow b$

Step 3: Restrict the number of variables on the RHS by two.

$S \rightarrow C_a\,A\,C_b\,B$, is replaced by $S \rightarrow C_a\,D_1$

$D_1 \rightarrow A\,D_2$

$D_2 \rightarrow C_b\,B$

$\therefore$ **The equivalent grammar in CNF $G' = (V', T, P, S)$**

Where $V' = \{S, A, B, D_1, D_2, C_a, C_b\}$ $T = \{a, b\}$ $S = S$

$P' = \{S \rightarrow C_a\,D_1$

$A \rightarrow C_a\,A | a$

$B \rightarrow C_b\,B | b$

$D_1 \rightarrow A\,D_2$

$D_2 \rightarrow C_b\,B$

$C_a \rightarrow a$

$C_b \rightarrow b$

Problem 7

Find a grammar in CNF equivalent to the grammar

$S \rightarrow \sim S \mid [\,S + S\,] \mid p \mid q$

where $T = \{\sim, [, +,], p, q\}$

SOLUTION

Step 1: No useless symbols, $\in$ - productions and unit productions.

$S \to p$

$S \to q$, already in CNF

Step 2: Replace terminals by variables.

$S \to \sim S$, is replaced by $S \to C_1 S$, $C_1 \to \sim$

$S \to [\, S + S\,]$, is replaced by $S \to C_2 S C_3 S C_4$

$C_2 \to [$

$C_3 \to +$

$C_4 \to]$

Step 3: Restrict the number of variables on the RHS by two.

$S \to C_2 S C_3 S C_4$, is replaced by $S \to C_2 D_1$

$D_1 \to S D_2$

$D_2 \to C_3 D_3$

$D_3 \to S C_4$

The equivalent grammar in CNF $G' = (V', T, P', S)$

Where $V' = \{S, D_1, D_2, D_3, C_1, C_2, C_3, C_4\}$

$T = \{\sim, [, +,], p, q\}$

$S = S$

$P' = \{S \to C_1 S \mid C_2 D_1 \mid p \mid q$

$D_1 \to S D_2 \quad D_2 \to C_3 D_3 \quad D_3 \to S C_4$

$C_1 \to \sim \quad C_2 \to [\quad C_3 \to + \quad C_4 \to]\,\}$

PROBLEM 8

Convert the given grammar to CNF

$S \to A B \mid C A$

$B \to B C \mid A B$

$A \to a$

$C \to a B \mid b$

SOLUTION

Step 1: No useless symbols, $\in$ - productions and unit productions.

$S \to A B \mid C A, A \to a$

$B \to B C \mid A B, C \to b$, already in CNF

Step 2: Replace terminals by variables $C \to a\ B$, is replaced by $C \to C_a\ B,\ C_a \to a$

Step 3: Not applicable

The equivalent grammar in CNF $G' = (V', T, P' P', S)$

where $V = \{S, A, B, C\}$ $\quad$ T = $\{a, b\}$ $\quad$ $S = S$

$P = \{S \to A\ B \mid C\ A$

$B \to B\ C \mid A\ B$

$A \to a$

$C \to C_a\ B \mid b$

$C_a \to a\ \}$

PROBLEM 9

Write the equivalent CNF, for the following grammar.

$S \to A\ B$

$A \to a\ a\ b$

$B \to a\ A\ C$

SOLUTION

Step 1: No useless symbols, $\in$ - productions and unit productions

$S \to A\ B$, already in CNF

Step 2: Replace terminals by variables.

$A \to a\ a\ b$, is replaced by $\quad A \to C_a\ C_a\ C_b$

$C_a \to a$

$C_b \to b$

$B \to a\ A\ C$, is replaced by $\quad B \to C_a\ A\ C,\ \ C_a \to a$

Step 3: Restrict the number of variables on the RHS by two

$A \to C_a\ C_a\ C_b$, is replaced by $\quad A \to C_a\ D_1,\quad D_1 \to C_a\ C_b$

$B \to C_a\ A\ C$, is replaced by $\quad B \to C_a\ D_2,\quad D_2 \to A\ C$

The equivalent grammar in CNF $G = (V, T, P, S)$

where $V = \{S, A, B, D_1, D_2, C_a, C_b\}$ $\quad$ T = $\{a, b\}$ $\quad$ $S = S$

$P = \{S \to A\ B$

$A \to C_a\ D_1$

$B \to C_a\ D_2$

$D_1 \to C_a\ C_b$

$D_2 \to A\ C$

$C_a \to a$

$C_b \to b\ \}$

Problem 10

educe the grammar, into CNF

$S \to A0B$

$A \to AA \mid 0S \mid 0$

$B \to 0BB \mid 1S \mid 1$

Solution

tep 1: No unit productions, $\in$ - productions and useless symbols.

$A \to AA \mid 0$

$B \to 1$, already in CNF

Step 2: Replace terminals by variables.

$S \to A0B$, is replaced by	$S \to AC_0B,\ C_0 \to 0$
$A \to 0S$, is replaced by	$A \to C_0S,\ C_0 \to 0$
$B \to 0BB$, is replaced by	$B \to C_0BB,\ C_0 \to 0$
$B \to 1S$, is replaced by	$B \to C_1S,\ C_1 \to 1$

Step 3: Restrict the number of variables on the RHS by two.

$S \to AC_0B$ is replaced by	$S \to AD_1,\ D_1 \to C_0B$
$B \to C_0BB$ is replaced by	$B \to C_0D_2,\ D_2 \to BB$

. **The equivalent grammar in CNF $G' = (V', T, P', S)$**

where $V' = \{S, A, B, D_1, D_2, C_0, C_1\}$

$T = \{0, 1\}$

$S = S$

$P' = \{S \to AD_1$

$A \to AA \mid 0 \mid C_0S$

$B \to C_1S \mid C_0D_2 \mid 1$

$D_1 \to CB$

$D_2 \to BB$

$C_0 \to 0$

$C_1 \to 1\}$

4.9 - Greibach Normal form (GNF)

In CNF there is restriction on the number of symbols on the right hand side of the production (i.e.,) it should be either two variables or one terminal.

In GNF there is no restriction on the length of RHS of a production but restrictions on the position in which terminals and variables appear.

Definition: GNF

Let $G = (V, T, P, S)$ be a CFG. The CFG is said to be in GNF if all the productions are of the form

$A \rightarrow a\,\alpha$ where $a \in T$ and $\alpha \in V^*$

(i.e.,) The first symbol on the right hand side of the production must be a terminal followed by zer or more number of variables.

Example 1:

The grammar

$$\begin{aligned} S &\rightarrow aA \mid bBB \mid bB \\ A &\rightarrow aA \mid bB \mid b \\ B &\rightarrow b \end{aligned}$$

is in GNF since the RHS of all the productions is a followed by zero on more variables terminal.

Example 2:

The grammar

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow abB \\ B &\rightarrow abb \end{aligned}$$

is not in GNF as the RHS of the production $S \rightarrow AB$, is not in the form $A \rightarrow a\,\alpha$ where $a \in T$ and $\alpha \in V$

4.9.1 Reduction of CFG to Greibach Normal Form

4.9.1.1 Theorem

Let $G = (V, T, P, S)$ be a CFG, generating the language L without $\in$. An equivalent grammar G generating the same language exists for which each production is of the form

$$A \rightarrow a\,\alpha$$

Step 1: Obtain the grammar in CNF

Step 2: Rename the non-terminals to A1, A2, A3, ...

Step 3: Using substitution method, obtain the production to the form

$$A_i \rightarrow A_j\alpha \text{ for } i < j$$

where $\alpha \in V^*$.

Step 4: After substitution, if the grammar has left recursion, eliminate left recursion.

Step 5: Repeat step 3 and|or step 4 to get the grammar in GNF.

PROBLEM 1

vert the given CFG to GNF

$S \rightarrow C A$

$A \rightarrow a$

$C \rightarrow a B \mid b$

SOLUTION

productions

$A \rightarrow a$

$C \rightarrow a B \mid b$ is already in GNF.

production $S \rightarrow C A$ is in CNF. It needs to be changed to GNF.

stitute the production $C \rightarrow a B \mid b$ into $S \rightarrow C A$

$S \rightarrow CA$

$\rightarrow a B A \mid b A$

ich is in GNF.

en we can write equivalent GNF as

$S \rightarrow a B A \mid b A$

$A \rightarrow a$

$C \rightarrow a B \mid b$

PROBLEM 2

nvert the following grammar into GNF

$S \rightarrow A B$

$A \rightarrow B S B$

$A \rightarrow a$

$B \rightarrow b$

SOLUTION

here are no useless symbols, no ∈ - productions, no unit productions.

ep 1: Transform G into CNF.

Following productions are already in CNF

$S \rightarrow A B$

$A \rightarrow a$

$B \rightarrow b$

The production $A \to B\,S\,B$ need to be converted to CNF.

$A \to B\,D_1$

$D_1 \to S\,B$

$\therefore$ Grammar G is in CNF

$S \to A\,B$

$A \to B\,D_1 | a$

$D_1 \to S\,B$

$B \to b$

Step 2: Rename the variables to $A_1, A_2, A_3, \ldots\ldots$

$S = A_1 ; A = A_2 ; B = A_3 ; D_1 = A_4$

Now the grammar can be re-written as

$A_1 \to A_2\,A_3$

$A_2 \to A_3\,A_4 \mid a$

$A_3 \to b$

$A_4 \to A_1\,A_3$

Step 3: Order the productions and remove left recursion where necessary.

The production $A_4 \to A_1\,A_3$ need to be changed since the production is not in the form

$A_i \to A_j\,\alpha$ for $i < j$

- Substitute $A_1 \to A_2 A_3$ in A_4 Production
  $A_4 \to A_2\,A_3\,A_3$
- Substitute $A_2 \to A_3 A_4 | a$ in A_4 Production
  $A_4 \to A_3\,A_4\,A_3\,A_3 \mid a\,A_3\,A_3$
- Substitute $A_3 \to b$ in A_4 Production
  $A_4 \to b\,A_4\,A_3\,A_3 \mid a\,A_3\,A_3$

All the productions are now properly ordered

$A_1 \to A_2\,A_3$

$A_2 \to A_3\,A_4 \mid a$

$A_3 \to b$

$A_4 \to b\,A_4\,A_3\,A_3 \mid a\,A_3\,A_3$

Step 4: Substitute backward to get GNF

- Substitute $A_3 \to b$ in A_2
  $A_2 \to b\,A_4 \mid a$
- Substitute $A_2 \to bA_4 | a$ in A_1 Production
  $A_1 \to b\,A_4\,A_3 \mid a\,A_3$

p 5: The final grammar in GNF

$A_1 \to b A_4 A_3 \mid a A_3$

$A_2 \to b A_4 \mid a$

$A_3 \to b$

$A_4 \to b A_4 A_3 A_2 \mid a A_3 A_3$

PROBLEM 3

nvert the following grammar into GNF

$S \to A$

$A \to a B a \mid a$

$B \to b A b \mid b$

SOLUTION

here are no useless symbols, no $\in$ - productions.

move unit production $S \to A$. By substituting A production into S.

$S \to a B a \mid a$

ep 1: Transform G into CNF G

(a) Substitute terminals on RHS with variables.

$S \to aBa\|a$ is replaced by	$S \to C_a B C_a \mid a$	$C_a \to a$
$A \to aBa\|a$ is replaced by	$A \to C_a B C_a \mid a$	$C_a \to a$
$B \to bBb\|b$ is replaced by	$B \to C_a A C_a \mid b$	$C_b \to b$

(b) Breakdown the rules to get the grammar G in CNF.

$S \to C_a D_1 \mid a \quad D_1 \to B C_a \qquad C_a \to a$

$A \to C_a D_1 \mid a$

$B \to C_b D_2 \mid b \quad D_2 \to A C_b \qquad C_b \to b$

tep 2: Rename the variables to $A_1, A_2, A_3, \ldots\ldots$

$S = A_1, \quad A = A_2, B = A_3, \quad C_a = A_4, C_b = A_5, \quad D_1 = A_6, D_2 = A_7$

$A_1 \to A_4 A_6 \mid a$

$A_2 \to A_4 A_6 \mid a$

$A_3 \to A_5 A_7 \mid b$

$A_6 \to A_3 A_4$

$A_7 \to A_2 A_5$

$A_4 \to a$

$A_5 \to b$

Step 3: Order the productions and remove recursion where necessary.

The productions

$A_6 \to A_3 A_4$

$A_7 \to A_2 A_5$

need to be changed since the productions are not in the form

$A_i \to A_j \alpha$ *for* $i < j$

$A_6 \to A_3 A_4$

- Substitute $A_3 \to A_5 A_7 | b$ in to A_6

  $A_6 \to A_5 A_7 A_4 \mid b A_4$
- Substitute $A_5 \to b$ in to A_6

  $A_6 \to bA_7 A_4 | bA_4$

  $A_7 \to A_2 A_5$
- Substitute $A_2 \to A_4 A_6 \mid a$ in to A_7

  $A_7 \to A_4 A_6 A_5 \mid a A_5$
- Substitute $A_4 \to a$ in to A_7

  $A_7 \to aA_6 A_5 | a A_5$

All the productions are now properly ordered.

$A_1 \to A_4 A_6 \mid a$

$A_2 \to A_4 A_6 \mid a$

$A_3 \to A_5 A_7 \mid b$

$A_4 \to a$

$A_5 \to b$

$A_6 \to b A_7 A_4 \mid b A_4$

$A_7 \to a A_6 A_5 \mid a A_5$

Step 4: Substitute backward to get GNF.

- Substitute $A_1 \to a$ in to A_1, A_2

  $A_1 \to a A_6 \mid a$

  $A_2 \to a A_6 \mid a$

  $A_3 \to A_5 A_7 \mid b$
- Substitute $A_5 \to a$ in A_3

  $A_3 \to b A_7 \mid b$

Step 5: The final grammar in GNF

$A_1 \to a A_6 \mid a$

$A_2 \rightarrow a\,A_6 \mid a$

$A_3 \rightarrow b\,A_7 \mid b$

$A_4 \rightarrow a$

$A_5 \rightarrow b$

$A_6 \rightarrow b\,A_7\,A_4 \mid b\,A_4$

$A_7 \rightarrow a\,A_6\,A_5 \mid a\,A_5$

Problem 4

Convert the following grammar in GNF

$S \rightarrow AB1 \mid 0$

$A \rightarrow 00A \mid B$

$B \rightarrow 1A1$

Solution

There are no useless symbols, no $\in$ - productions.

Remove unit production $A \rightarrow B$ by substituting B into A.

$A \rightarrow 0\,0\,A \mid 1\,A\,1$

Step 1: Transform G into CNF G .

(a) Substitute terminals on RHS with variables.

$S \rightarrow AB1 \mid 0$ replaced by,	$S \rightarrow A\,B\,C_1 \mid 0$	$C_1 \rightarrow 1$
$A \rightarrow 00A \mid 1A1$ replaced by,	$A \rightarrow C_0\,C_0\,A \mid C_1\,AC_1$	$C_0 \rightarrow 0$
$B \rightarrow 1A1$ replaced by,	$B \rightarrow C_1\,A\,C_1$	

(b) Breakdown the rules to get the grammar G in CNF.

$S \rightarrow ABC_1 \mid 0$ replaced by	$S \rightarrow A\,D_1 \mid 0$	$D_1 \rightarrow B\,C_1$
$A \rightarrow C_0C_0A$ replaced by	$A \rightarrow C_0\,D_2$	$D_2 \rightarrow C_0A$
$A \rightarrow C_1AC_1$ replaced by	$A \rightarrow C_1\,D_3$	$D_3 \rightarrow A\,C_1$
$B \rightarrow C_1AC_1$ replaced by	$B \rightarrow C_1\,D_3$	

$C_0 \rightarrow 0$

$C_1 \rightarrow 1$

Step 2: Rename the variables to $A_1, A_2, A_3 \ldots\ldots$

$S = A_1, A = A_2, B = A_3, C_0 = A_4, C_1 = A_5, D_1 = A_6, D_2 = A_7, D_3 = A_8$

$A_1 \rightarrow A_2\,A_6 \mid 0$

$A_2 \rightarrow A_4\,A_7 \mid A_5\,A_8$

$A_3 \rightarrow A_5\,A_8$

$A_4 \rightarrow 0$

$A_5 \rightarrow 1$

$A_6 \rightarrow A_3 A_5$

$A_7 \rightarrow A_4 A_2$

$A_8 \rightarrow A_2 A_5$

Step 3: Order the productions and remove recursion where necessary

The productions

$A_6 \rightarrow A_3 A_5$

$A_7 \rightarrow A_4 A_2$

$A_8 \rightarrow A_2 A_5$

need to be changed since the productions are not in the form

$A_i \rightarrow A_j \alpha \; for \; i < j$

$A_6 \rightarrow A_3 A_5$

- Substitute $A_6 \rightarrow A_5 A_8 A_5$ into A_6

  $A_6 \rightarrow A_5 A_8 A_5$
- Substitute $A_5 \rightarrow 1$ into A_6

  $A_6 \rightarrow 1 A_8 A_5$

  $A_7 \rightarrow A_4 A_2$
- Substitute $A_4 \rightarrow 0$ into A_7

  $A_7 \rightarrow 0 A_2$

  $A_8 \rightarrow A_2 A_5$
- Substitute $A_2 \rightarrow A_4 A_7 \mid A_5 A_8$ into A_8

  $A_8 \rightarrow A_4 A_7 A_5 \mid A_5 A_8 A_5$
- Substitute $A_4 \rightarrow 0 \quad A_5 \rightarrow 1$ into A_8

  $A_8 \rightarrow 0 A_7 A_5 \; A_8 A_5$

All the productions are now properly ordered.

$A_1 \rightarrow A_2 A_6 \mid 0$

$A_2 \rightarrow A_4 A_7 \mid A_5 A_8$

$A_3 \rightarrow A_5 A_8$

$A_4 \rightarrow 0$

$A_5 \rightarrow 1$

$A_6 \rightarrow 1 A_8 A_5$

$A_7 \rightarrow 0 A_2$

$A_8 \rightarrow 0 A_7 A_5 \mid 1 A_8 A_5$

ep 4: Substitute backward to get GNF

- Substitute $A_5 \to A_3$
  $A_3 \to 1A_8$ into A_6
- Substitute A_4, A_5 in A_2
  $A_2 \to 0\, A_7\, A_5 \mid 1\, A_8$
- Substitute A_2 in A_1
  $A_1 \to (0\, A_7 \mid 1\, A_8)\, A_6 \mid 0$
  $A_1 \to 0\, A_7\, A_6 \mid 1\, A_8\, A_6 \mid 0$

ep 5: The final grammar in GNF

$A_1 \to 0\, A_7\, A_6 \mid 1\, A_8\, A_6 \mid 0$
$A_2 \to 0\, A_7 \mid 1\, A_8$
$A_3 \to 1\, A_8$
$A_4 \to 0$
$A_5 \to 1$
$A_6 \to 1\, A_8\, A_5$
$A_7 \to 0\, A_2$
$A_8 \to 0\, A_7\, A_5 \mid 1\, A_8\, A_5$

PROBLEM 5

Convert the following grammar into GNF

$A \to B\, C$
$B \to C\, A \mid b$
$C \to A\, B \mid a$

SOLUTION

Step 1: The grammar is already in CNF.

Step 2: Rename the variables to $A_1, A_2, A_3 \ldots\ldots$

Let $A = A_1, B = A_2, C = A_3$

$A_1 \to A_2\, A_3$
$A_2 \to A_3\, A_1 \mid b$
$A_3 \to A_1\, A_2 \mid a$

Step 3: Order the productions and remove recursion where necessary.

The productions $A_3 \to A_1\, A_2$ need to changed since the productions are not in the form

$A_i \to A_j\, \alpha$ for $i < j$
$A_3 \to A_1\, A_2 \mid a$

- Substitute $A_1 \rightarrow A_2 A_3$ into A_3

  $A_3 \rightarrow A_2 A_3 A_2 \mid a$
- Substitute $A_2 \rightarrow A_3 A_1 \mid b$ into A_3

  $A_3 \rightarrow A_3 A_1 A_3 A_2 \mid bA_3 A_2 \mid a$

Since A_3 production is having left recursion, eliminate it.

$A_3 \rightarrow b A_3 A_2 \mid aZ$

$Z \rightarrow A_1 A_3 A_2 Z \mid \in$

eliminate $\in$ production, by eliminating nullable variable Z in the productions

$A_3 \rightarrow b A_3 A_2 Z \mid b A_3 A_2 \mid a Z \mid a$

$Z \rightarrow A_1 A_3 A_2 Z \mid A_1 A_3 A_2$

Step 4: Since all the productions are properly ordered. Substitute backward to get GNF

- Substitute A_3 in A_2 production

  $A_2 \rightarrow A_3 A_1 \mid b$

  $A_2 \rightarrow (b A_3 A_2 Z \mid b A_3 A_2 \mid a Z \mid a) A_1 \mid b$

  $A_2 \rightarrow b A_3 A_2 Z A_1 \mid b A_3 A_2 A_1 \mid a Z A_1 \mid a A_1 \mid b$
- Substitute A_2 in A_1 production

  $A_1 \rightarrow A_2 A_3$

  $A_1 \rightarrow (b A_3 A_2 Z A_1 \mid b A_3 A_2 A_1 \mid a Z A_1 \mid a A_1 \mid b) A_3$

  $A_1 \rightarrow b A_3 A_2 Z A_1 A_3 \mid b A_1 A_2 A_1 A_3 \mid a Z A_1 A_3 \mid a A_1 A_3 \mid b A_3$
- Substitute A_1 in Z production

  $Z \rightarrow A_1 A_3 A_2 \mid A_1 A_3 A_2 Z$

  $\rightarrow (b A_3 A_2 Z A_1 A_3 \mid b A_3 A_2 A_1 A_3 \mid a Z A_1 A_3 \mid a A_1 A_3 \mid b A_3) A_3 A_2 \mid$

  $(b A_3 A_2 Z A_1 A_3 \mid b A_3 A_2 A_1 A_3 \mid a Z A_1 A_3 \mid a A_1 A_3 \mid b A_3) A_3 A_2 Z$

  $Z \rightarrow b A_3 A_2 Z A_1 A_3 A_3 A_2 \mid b A_1 A_2 A_1 A_3 A_3 A_2 \mid a A_1 A_3 A_3 A_2 \mid a z A_1 A_3 A_3 A$

  $b A_3 A_3 A_2 \mid b A_3 A_2 Z A_1 A_3 A_3 A_2 Z \mid b A_3 A_2 A_1 A_3 A_3 A_2 Z \mid a A_1 A_3 A_3 A_2 Z \mid$

  $a Z A_1 A_3 A_3 A_2 Z \mid b A_3 A_3 A_2 Z$

Step 5: The final GNF is

$A_1 \rightarrow b A_3 A_2 A_1 A_3 \mid a A_1 A_3 \mid b A_3 A_2 Z A_1 A_3 \mid a Z A_1 A_3 \mid b A_3$

$A_2 \rightarrow b A_3 A_2 A_1 \mid a A_1 \mid b A_3 A_2 Z A_1 \mid a Z A_1 \mid b$

$A_3 \rightarrow b A_3 A_2 \mid a \mid b A_3 A_2 Z \mid a Z$

$Z \rightarrow b A_3 A_2 Z A_1 A_3 A_3 A_2 \mid b A_3 A_2 A_1 A_3 A_3 A_2 \mid a A_1 A_3 A_3 A_2 \mid a z A_1 A_3 A_3 A$

$b A_3 A_3 A_2 \mid b A_3 A_2 Z A_1 A_3 A_3 A_2 Z \mid b A_3 A_2 A_1 A_3 A_3 A_2 Z \mid a A_1 A_3 A_3 A_2 Z \mid$

$a z A_1 A_3 A_3 A_2 z \mid b A_3 A_3 A_2 Z$

- Substitute $A_1 \to A_2 A_3$ into A_3

  $A_3 \to A_2 A_3 A_2 \mid a$
- Substitute $A_2 \to A_3 A_1 \mid b$ into A_3

  $A_3 \to A_3 A_1 A_3 A_2 \mid bA_3 A_2 \mid a$

Since A_3 production is having left recursion, eliminate it.

$A_3 \to b A_3 A_2 \mid aZ$

$Z \to A_1 A_3 A_2 Z \mid \in$

eliminate $\in$ production, by eliminating nullable variable Z in the productions

$A_3 \to b A_3 A_2 Z \mid b A_3 A_2 \mid a Z \mid a$

$Z \to A_1 A_3 A_2 Z \mid A_1 A_3 A_2$

Step 4: Since all the productions are properly ordered. Substitute backward to get GNF

- Substitute A_3 in A_2 production

  $A_2 \to A_3 A_1 \mid b$

  $A_2 \to (b A_3 A_2 Z \mid b A_3 A_2 \mid a Z \mid a) A_1 \mid b$

  $A_2 \to b A_3 A_2 Z A_1 \mid b A_3 A_2 A_1 \mid a Z A_1 \mid a A_1 \mid b$
- Substitute A_2 in A_1 production

  $A_1 \to A_2 A_3$

  $A_1 \to (b A_3 A_2 Z A_1 \mid b A_3 A_2 A_1 \mid a Z A_1 \mid a A_1 \mid b) A_3$

  $A_1 \to b A_3 A_2 Z A_1 A_3 \mid b A_3 A_2 A_1 A_3 \mid a Z A_1 A_3 \mid a A_1 A_3 \mid b A_3$
- Substitute A_1 in Z production

  $Z \to A_1 A_3 A_2 \mid A_1 A_3 A_2 Z$

  $\to (b A_3 A_2 Z A_1 A_3 \mid b A_3 A_2 A_1 A_3 \mid a Z A_1 A_3 \mid a A_1 A_3 \mid b A_3) A_3 A_2 \mid$

  $(b A_3 A_2 Z A_1 A_3 \mid b A_3 A_2 A_1 A_3 \mid a Z A_1 A_3 \mid a A_1 A_3 \mid b A_3) A_3 A_2 Z$

  $Z \to b A_3 A_2 Z A_1 A_3 A_3 A_2 \mid b A_3 A_2 A_1 A_3 A_3 A_2 \mid a A_1 A_3 A_3 A_2 \mid a z A_1 A_3 A_3 A$

  $b A_3 A_3 A_2 \mid b A_3 A_2 Z A_1 A_3 A_3 A_2 Z \mid b A_3 A_2 A_1 A_1 A_3 A_2 Z \mid a A_1 A_3 A_3 A_2 Z$

  $a Z A_1 A_3 A_3 A_2 Z \mid b A_3 A_3 A_2 Z$

Step 5: The final GNF is

$A_1 \to b A_3 A_2 A_1 A_3 \mid a A_1 A_3 \mid b A_3 A_2 Z A_1 A_3 \mid a Z A_1 A_3 \mid b A_3$

$A_2 \to b A_3 A_2 A_1 \mid a A_1 \mid b A_3 A_2 Z A_1 \mid a Z A_1 \mid b$

$A_3 \to b A_3 A_2 \mid a \mid b A_3 A_2 Z \mid a Z$

$Z \to b A_3 A_2 Z A_1 A_3 A_3 A_2 \mid b A_3 A_2 A_1 A_3 A_3 A_2 \mid a A_1 A_3 A_3 A_2 \mid a z A_1 A_3 A_3 A$

$b A_3 A_3 A_2 \mid b A_3 A_2 Z A_1 A_3 A_3 A_2 Z \mid b A_3 A_2 A_1 A_3 A_3 A_2 Z \mid a A_1 A_3 A_3 A_2 Z \mid$

$a z A_1 A_3 A_3 A_2 z \mid b A_3 A_3 A_2 Z$

Problem 6

Convert the following grammar in GNF

$S \to AA \mid 0$

$A \to SS \mid 1$

Solution

Step 1: The grammar is already in GNF

Step 2: Rename the variables as $A_1, A_2, A_3, \ldots\ldots$

Let $S = A_1, A_1 = A_2$

$A_1 \to A_2 A_2 \mid 0$

$A_2 \to A_1 A_1 \mid 1$

Step 3: Order the productions and remove recursion where necessary.

The production

$A_2 \to A_1 A_1$

need to be changed since the productions are not in the form

$A_i \to A_j \alpha \qquad i < j$

- Substitute A_1 in A_2

  $A_2 \to A_1 A_1 | 1$

  $A_2 \to (A_2 A_2 \mid 0) A_1 \mid 1$

  $A_2 \to A_2 A_2 A_1 \mid 0 A_1 \mid 1$

A_2 production is having left recursion eliminate it.

$A_2 \to (0 A_1 \mid 1) Z$

$Z \to A_2 A_1 Z \mid \epsilon$

Eliminate ϵ production by eliminating nullable variable Z in the productions.

$A_2 \to 0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1$

$Z \to A_2 A_1 Z \mid A_2 A_1$

Step 4: Since all the productions are properly ordered, substitute backward to get GNF.

- Substitute A_2 into A_1

  $A_1 \to A_2 A_2 | 0$

  $A_1 \to (0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1) A_2 \mid 0$

  $A_1 \to 0 A_1 Z A_2 \mid 0 A_1 A_2 \mid 1 Z A_2 \mid 1 A_2 \mid 0$

- Substitute A_2 into Z

  $Z \to A_2 A_1 Z \mid A_2 A_1$

$$Z \rightarrow (0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1) A_1 Z \mid (0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1) A_1$$

$$Z \rightarrow 0 A_1 Z A_1 Z \mid 0 A_1 A_1 Z \mid 1 Z A_1 Z \mid 1 A_1 Z \mid 0 A_1 Z A_1 \mid 0 A_1 A_1 \mid 1 Z A_1 \mid 1 A_1$$

Step 5: The final GNF is given below

$$A_1 \rightarrow 0 A_1 Z A_2 \mid 0 A_1 A_2 \mid 1 Z A_2 \mid 1 A_2 \mid 0$$

$$A_2 \rightarrow 0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1$$

$$Z \rightarrow 0 A_1 Z A_1 Z \mid 0 A_1 A_1 Z \mid 1 Z A_1 Z \mid 1 A_1 Z \mid 0 A_1 Z A_1 \mid 0 A_1 A_1 \mid 1 Z A_1 \mid 1 A_1$$

4.10 CONVERSION OF CFG TO PDA

Conversion of CFG to PDA is possible only if the CFG is in GNF. The steps to be followed to convert a grammar to its equivalent PDA are shown below.

1. Convert the grammar to GNF.
2. In state q_0 when z_0 is on the top of the stack, without processing any input symbol, push the start symbol S on to the stack and change the state to q_1.
   $\delta (q_0, \in, z_0) = (q_1, S z_0)$
3. For every production of the form $A \rightarrow a\, \alpha$ where $\alpha \in V^*$ introduce the transition
   $\delta (q_1, a, A) = (q_1, \alpha)$
4. Finally in state q_1, without processing any input symbol change the state to q_2, which is an accepting state.
   $\delta (q_1, \in, z_0) = (q_2, z_0)$

PROBLEM 1

For the grammar

$$S \rightarrow a ABC$$
$$A \rightarrow aB \mid a$$
$$B \rightarrow bA \mid b$$
$$C \rightarrow a$$

obtain the equivalent PDA.

SOLUTION

The given CFG is already in GNF.

Let q_0 be the start state and z_0 be the initial symbol on the stack.

Step 1: Push the start symbol S on to the stack and change the state to q_1. The transition of this can be of the form

$$\delta (q_0, \in, z_0) = (q_1, Sz_0)$$

Step 2: For each production $A \rightarrow a\, \alpha$ where $\alpha \in v^*$ introduce the transition

$$\delta (q_1, a, A) = (q_1, \alpha)$$

This can be done as shown below.

$$Z \to (0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1) A_1 Z \mid (0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1) A_1$$

$$Z \to 0 A_1 Z A_1 Z \mid 0 A_1 A_1 Z \mid 1 Z A_1 Z \mid 1 A_1 Z \mid 0 A_1 Z A_1 \mid 0 A_1 A_1 \mid 1 Z A_1 \mid 1 A_1$$

Step 5: The final GNF is given below

$$A_1 \to 0 A_1 Z A_2 \mid 0 A_1 A_2 \mid 1 Z A_2 \mid 1 A_2 \mid 0$$

$$A_2 \to 0 A_1 Z \mid 0 A_1 \mid 1 Z \mid 1$$

$$Z \to 0 A_1 Z A_1 Z \mid 0 A_1 A_1 Z \mid 1 Z A_1 Z \mid 1 A_1 Z \mid 0 A_1 Z A_1 \mid 0 A_1 A_1 \mid 1 Z A_1 \mid 1 A_1$$

4.10 CONVERSION OF CFG TO PDA

Conversion of CFG to PDA is possible only if the CFG is in GNF. The steps to be followed to convert a grammar to its equivalent PDA are shown below.

1. Convert the grammar to GNF.
2. In state q_0 when z_0 is on the top of the stack, without processing any input symbol, push the start symbol S on to the stack and change the state to q_1.

   $\delta(q_0, \in, z_0) = (q_1, S z_0)$
3. For every production of the form $A \to a\,\alpha$ where $\alpha \in V^*$ introduce the transition

   $\delta(q_1, a, A) = (q_1, \alpha)$
4. Finally in state q_1, without processing any input symbol change the state to q_2, which is an accepting state.

   $\delta(q_1, \in, z_0) = (q_2, z_0)$

PROBLEM 1

For the grammar

$$S \to aABC$$
$$A \to aB \mid a$$
$$B \to bA \mid b$$
$$C \to a$$

obtain the equivalent PDA.

SOLUTION

The given CFG is already in GNF.

Let q_0 be the start state and z_0 be the initial symbol on the stack.

Step 1: Push the start symbol S on to the stack and change the state to q_1. The transition of this can be of the form

$\delta(q_0, \in, z_0) = (q_1, Sz_0)$

Step 2: For each production $A \to a\,\alpha$ where $\alpha \in v^*$ introduce the transition

$\delta(q_1, a, A) = (q_1, \alpha)$

This can be done as shown below.

Production	Transition
$S \rightarrow a\,ABC$	$\delta(q_1, a, S) = (q_1, ABC)$
$A \rightarrow a\,B$	$\delta(q_1, a, A) = (q_1, B)$
$A \rightarrow a$	$\delta(q_1, a, A) = (q_1, \in)$
$B \rightarrow b\,A$	$\delta(q_1, b, B) = (q_1, A)$
$B \rightarrow b$	$\delta(q_1, b, B) = (q_1, \in)$
$C \rightarrow a$	$\delta(q_1, a, C) = (q_1, \in)$

Step 3: Finally in state q_1, without consuming any input symbol change the state to q_2 which is an accepting state.

$\delta(q_1, \in, z_0) = (q_2, z_0)$

Step 4:

The PDA M for the given CFG can be defined as

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

where $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Gamma = \{S, A, B, C, z_0\}$

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{q_2\}$

δ: is shown below

$\delta(q_0, \in, z_0) = (q_0, S\,z_0)$	$\delta(q_1, b, B) = (q_1, A)$
$\delta(q_0, a, S) = (q_1, ABC)$	$\delta(q_1, b, B) = (q_1, \in)$
$\delta(q_1, a, A) = (q_1, B)$	$\delta(q_1, a, C) = (q_1, \in)$
$\delta(q_1, a, A) = (q_1, \in)$	$\delta(q_1, \in, z_0) = (q_2, z_0)$

PROBLEM 2

Obtain the equivalent PDA for the given CFG.

$S \rightarrow aAA$

$A \rightarrow aB$

$B \rightarrow b$

$B \rightarrow b$

SOLUTION

The given grammar is already in GNF.

Let q_0 be the start state and Z_0 be the initial symbol on the stack.

Step 1: Push the start symbol S on to the stack and change the state to q_1. The transition of this can be given as $\delta(q_0, \in, z_0) = (q_1, S z_0)$

Step 2: For each production $A \to a\,\alpha$ where $\alpha \in V^*$ introduce the transition

$\delta(q_1, a, A) = (q_1, \alpha)$

Production	Transition
$S \to a\,AA$	$\delta(q_1, a, S) = (q_1, AA)$
$A \to a\,B$	$\delta(q_1, a, A) = (q_1, B)$
$A \to b$	$\delta(q_1, b, A) = (q_1, \in)$
$B \to b$	$\delta(q_1, b, B) = (q_1, \in)$

Step 3: Finally in state q_1, without consuming any input change the state to q_2 which is an accepting state.

$\delta(q_1, \in, z_0) = (q_2, z_0)$

Step 4:

The PDA M for the given CFG cn be defined as

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Where	δ: is shown below
$Q = \{q_0, q_1, q_2\}$	$\delta(q_0, \in, z_0) = (q_1, S z_0)$
$\Sigma = \{a, b\}$	$\delta(q_1, a, S) = (q_1, AA)$
$\Gamma = \{S, A, B, z_0\}$	$\delta(q_1, a, A) = (q_1, B)$
$q_0 = \{q_0\}$	$\delta(q_1, b, A) = (q_1, \in)$
$z_0 = \{z_0\}$	$\delta(q_1, b, B) = (q_1, \in)$
$F = \{q_2\}$	$\delta(q_1, \in, z_0) = (q_2, z_0)$

PROBLEM 3

Construct the PDA for the grammar $S \to a\,S\,bb \mid a$

SOLUTION

The grammar is not in GNF. The GNF of this grammar is

$S \to a\,S\,A \mid a$

$A \to b\,B$

$B \to b$

Step 1: Push the start symbol S on to the stack and change the state to q_1. The transition of this can be given as,

$\delta(q_0, \in, z_0) = (q_1, S z_0)$

ep 2: For each production $A \rightarrow a\ \alpha$ where $\alpha \in V^*$ introduce the transition

$\delta\ (q_1, a, A) = (q_1, \alpha\ )$

Production	Transition
$S \rightarrow a\ S\ A$	$\delta\ (q_1, a, S) = (q_1, SA)$
$S \rightarrow a$	$\delta\ (q_1, a, S) = (q_1, \in)$
$A \rightarrow b\ B$	$\delta\ (q_1, b, A) = (q_1, B)$
$B \rightarrow b$	$\delta\ (q_1, b, B) = (q_1, \in)$

ep 3: In state q , without consuming any input change the state to q_2 which is an accepting state.

$\delta\ (q_1, \in, z_0) = (q_2, z_0)$

tep 4:

The PDA M for the given GFG is

$M\ = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

Where	δ: is shown below
$Q\ = \{q_0, q_1, q_2\ \}$	$\delta\ (q_1, \in, z_0) = (q_1, Sz_0)$
$\Sigma\ = \{\ a, b\ \}$	$\delta\ (q_1, a, S) = (q_1, SA)$
$\Gamma\ = \{\ S, A, B, z_0\ \}$	$\delta\ (q_1, a, S) = (q_1, \in)$
$q_0\ = \{\ q_0\ \}$	$\delta\ (q_1, b_1, A) = (q_1, B)$
$z_0\ = \{\ z_0\ \}$	$\delta\ (q_1, b, B) = (q_1, \in)$
$F\ = \{\ q_2\ \}$	$\delta\ (q_1, \in, z_0) = (q_1, z_0)$

PROBLEM 4

btain a PDA to accept the language $L = \{\ w\ w^R \mid |w| \geq 1$ for $w \in (a + b)^*\ \}$

SOLUTION

h. equivalent CFG to generate the language is

$S \rightarrow a\ S\ A \mid a\ a$

$A \rightarrow b\ S\ b \mid b\ b$

· grammar is not in GNF The GNF of this grammar is

$S \rightarrow a\ S\ A \mid a\ A$

$S \rightarrow b\ S\ B \mid b\ B$

$A \rightarrow a$

$B \rightarrow b$

Step 1: Push the start symbol S on to the stack and change the state to q_1. $\delta(q_1, \in, z_0) = (q$

Step 2: The equivalent transitions for the given productions are

Production	Transition
$S \to aSA$	$\delta(q_1, a, S) = (q_1, SA)$
$S \to aA$	$\delta(q_1, a, S) = (q_1, A)$
$A \to a$	$\delta(q_1, a, A) = (q_1, \in)$
$B \to b$	$\delta(q_1, b, B) = (q_1, \in)$

Step 3: In state q_1, without consuming any input change the state to q_2 which is an accepting state.

$\delta(q_0, \in, z_0) = (q_2, z_0)$

Step 4: The PDA M for the given CFG can be defined as $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Where

$Q = \{q_0, q_1, q_2\}$	δ: is shown below
$\Sigma = \{a, b\}$	$\delta(q_0, \in, z_0) = (q_1, S z_0)$
$\Gamma = \{S, A, B, z_0\}$	$\delta(q_1, a, S) = (q_1, SA)$
$q_0 = \{q_0\}$	$\delta(q_1, a, S) = (q_1, A)$
$z_0 = \{z_0\}$	$\delta(q_1, a, A) = (q_1, \in)$
$F = \{q_2\}$	$\delta(q_1, b, B) = (q_1, \in)$
	$\delta(q_1, \in, z_0) = (q_2, z_0)$

Problem 6

Construct a PDA equivalence the following grammar.

$S \to aA$

$A \to aABC \mid bB \mid a$

$B \to b$

$C \to c$

Check whether the string *aaabc* is accepted by the PDA.

Solution

The given CFG is already in GNF Let q_0 be the start state and z_0 be the initial stack symbol.

Step 1: Push the start symbol. S on to the stack and move to state q_1. The transitions is

$\delta(q_0, \in, z_0) = (q_1, Sz_0)$

tep 2:

Production	Transition
$S \rightarrow aA$	$\delta(q_1, a, S) = (q_1, A)$
$A \rightarrow aABC$	$\delta(q_1, a, A) = (q_1, ABC)$
$A \rightarrow bB$	$\delta(q_1, b, A) = (q_1, B)$
$A \rightarrow a$	$\delta(q_1, a, A) = (q_1, \in)$
$B \rightarrow b$	$\delta(q_1, b, B) = (q_1, \in)$
$C \rightarrow c$	$\delta(q_1, c, C) = (q_1, \in)$

tep 3: $\delta(q_1, \in, z_0) = (q_2, z_0)$

tep 4:

The equivalent PDA, M is given by

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

Where

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b, c\}$

$\Gamma = \{Z_0, S, A, B, C\}$

$q_0 = \{ q_0 \}$

$z_0 = \{ z_0 \}$

$F = \{ q_2 \}$

δ is given below

$\delta(q_0, \in, z_0) = (q_1, Sz_0)$

$\delta(q_1, a, S) = (q_1, A)$

$\delta(q_1, a, A) = (q_1, ABC) \mid (q_1, \in)$

$\delta(q_1, b, A) = (q_1, B)$

$\delta(q_1, b, B) = (q_1, \in)$

$\delta(q_1, c, C) = (q_1, \in)$

$\delta(q_1, \in, z_0) = (q_2, z_0)$

check the acceptability to the string *aaabc*.

erivation of aaabc

$$\begin{aligned} S &\rightarrow aA \\ &\rightarrow aaABC \\ &\rightarrow aaaBC \\ &\rightarrow aaabC \\ &\rightarrow aaabc \end{aligned}$$

ecking the acceptability by the PDA.

$$\begin{aligned} (q_0, \in, z_0) &\vdash (q_1, aaabc, Sz_0) \\ &\vdash (q_1, aabc, Az_0) \\ &\vdash (q_1, abc, ABCz_0) \\ &\vdash (q_1, bc, BCz_0) \\ &\vdash (q_1, c, Cz_0) \\ &\vdash (q_1, \in, z_0) \end{aligned}$$

$\vdash (q_2, \in, z_0)$

$(q_2, \in, z_0)$ is the final configuration.

the PDA accepts the string *aaabc*.

Problem 7

Convert the grammer into a PDA and check whether the string *aaa* is accepted by the PDA.

$$S \to aAA$$
$$A \to aS|bS|a$$

Solution

The given CFG is already is GNF.

Let q_0 be the start state and z_0 be the initial stack symbol.

Step 1: $\delta(q_0, \in, z_0) = (q_1, Sz_0)$

Step 2:

Production	Transition
$S \to aAA$	$\delta(q_1, a, S) = (q_1, AA)$
$A \to aS$	$\delta(q_1, a, A) = (q_1, S)$
$A \to bS$	$\delta(q_1, b, A) = (q_1, S)$
$A \to a$	$\delta(q_1, a, A) = (q_1, \in)$

Step 3: $\delta(q_1, \in, z_0) = (q_2, z_0)$

Step 4:

the equivalent PDA, M is given by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

Where

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Gamma = \{z_0, S, A\}$

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{q_2\}$

δ is given below

$\delta(q_0, \in, z_0) = (q_0, Sz_0)$

$\delta(q_1, a, S) = (q_1, AA)$

$\delta(q_1, a, A) = (q_1, S)|(q_1, \in)$

$\delta(q_1, b, A) = (q_1, S)$

$\delta(q_1, b, A) = (q_1, S)$

$\delta(q_1, \in, z_0) = (q_2, z_0)$

To check the acceptability of the string *aaa*

Derivation of *aaa*

$$S \to aAA$$

$\rightarrow aaA$

$\rightarrow aaa$

ecking the acceptability by the PDA

$(q_0, \in, z_0) \vdash (q_1, aaa, Sz_0)$

$\vdash (q_1, aa, AAz_0)$

$\vdash (q_1, a, Az_0)$

$\vdash (q_1, \in, z_0)$

$\vdash (q_2, \in, z_0)$

is the final configuration

the PDA accepts the string aaa.

4.11 Pumping Lemma for Context free Languages

e pumping lemma for CFL s is useful for

a) Generating an infinite number of strings from a given sufficiently long string.

b) To prove that certain languages are not context free.

Theorem

t L be any CFL, then there exists an integer m such that, for any string $w \in$ L with $|w| > m$, the string w can be split into five parts as $w = u\,v\,x\,y\,z$ such that

(a) $|\,v\,y\,| \geq 1$

(b) $|\,v\,x\,y\,| \leq m$

(c) For all $i \geq 0$ $\quad u\,v^i\,x\,y^i\,z \in$ L

Proof

nsider an infinite context free grammar G with no unit productions and no $\in$-productions Consider a string $w \in$ L (G) with length, m where

$m >$ (number of productions) × (largest right side of productions)

is suggests that some variable must be repeated in the derivation of w

Consider u, v, x, y, z : string of terminals with

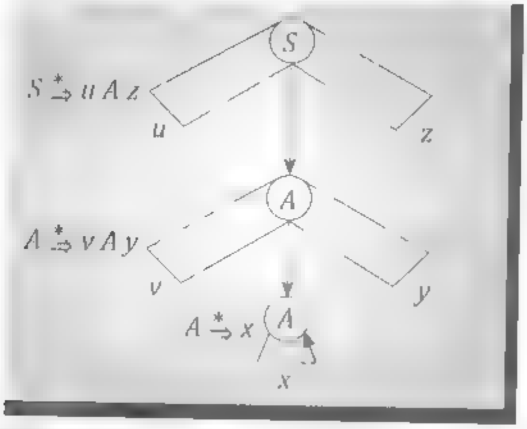
$S \rightarrow u\,A\,z$

$A \rightarrow v\,A\,y$

$A \rightarrow x$

and $w = u\,v\,x\,y\,z$

The tree of w is


$S \overset{*}{\Rightarrow} u A z$
S
u
z
$A \overset{*}{\Rightarrow} v A y$
A
v
y
$A \overset{*}{\Rightarrow} x$
A
x

Following are the strings that can be generated

(a) We know that

$S \overset{*}{\Rightarrow} u A z, A \overset{*}{\Rightarrow} v A y$ and $A \overset{*}{\Rightarrow} x$

$S \overset{*}{\Rightarrow} u A z, \overset{*}{\Rightarrow} u x z$

$\therefore \quad w = u v^0 x y^0 z$

(b) We know that

$S \overset{*}{\Rightarrow} u A z, A \overset{*}{\Rightarrow} v A y$ and $A \overset{*}{\Rightarrow} x$

$S \overset{*}{\Rightarrow} u A z, \overset{*}{\Rightarrow} u v A y z \overset{*}{\Rightarrow} u v x y z$

. $\quad w = u v^1 x y^1 z$

(c) We know that

$S \overset{*}{\Rightarrow} u A z, A \overset{*}{\Rightarrow} v A y$ and $A \overset{*}{\Rightarrow} x$

$S \overset{*}{\Rightarrow} u A z, \overset{*}{\Rightarrow} u v A y z \overset{*}{\Rightarrow} u v A y y z \overset{*}{\Rightarrow} u v v x y y z$

$w = u v^2 x y^2 z$

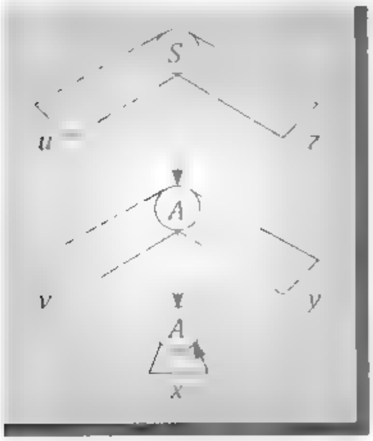
In general

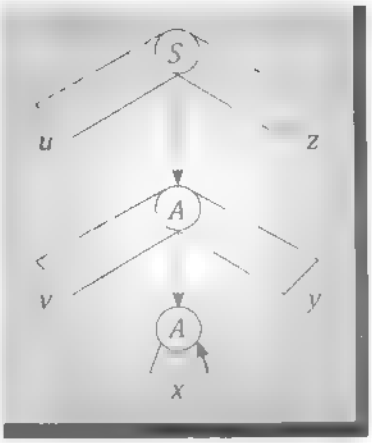
For productions $S \overset{*}{\Rightarrow} u A z, A \overset{*}{\Rightarrow} v A y$ and $A \overset{*}{\Rightarrow} x$.

The string generated is

$$\begin{aligned} S \; & \overset{*}{\Rightarrow} u A y, \overset{*}{\Rightarrow} u v A y z \overset{*}{\Rightarrow} u v v A y y z \overset{*}{\Rightarrow} \\ & \overset{*}{\Rightarrow} u v v v A y y y z \overset{*}{\Rightarrow} \ldots\ldots \\ & \overset{*}{\Rightarrow} u v v v \ldots\ldots v A y y y \ldots\ldots\ldots y z \\ & \overset{*}{\Rightarrow} u v v v \ldots\ldots v x y y y \ldots\ldots\ldots y z \\ & \overset{*}{\Rightarrow} u v^i x y^i z \end{aligned}$$

Therefore, a string of the form $u v^i x y^i z$ for $i \geq 0$ is generated by G. Further $u v x y z \in L$ (G implies that, $u v^i x y^i z \in L(G)$ with $| v x y | \leq m$ and $| v y | \geq 1$.





$| v x y | \leq m$, since A is the last repeated variable.

$| v y | \geq 1$ since there are no unit and $\in$ - production.

Problem 1

how that L = $\{a^n b^n c^n \mid n >, 1\}$ is not context free

Solution

en the grammar $L = \{ a^n b^n c^n \}$ we note that, in every string of L, any symbol appears the same number of times as any other symbol. Also a cannot appear after b and c cannot ppear before b an so on.

tep 1: Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

tep 2: Let $z = a^n b^n c^n$ Then $| z | = 3n > n$ Write $z = u\,v\,w\,x\,y$ where $| vx | > 1$ (i.e.,) atleast one of v or x not $\in$.

tep 3: $u\,v\,w\,x\,y = a^n b^n c^n$. As $1 \leq | v\,x | \leq n$, v or x cannot contain all the three symbols a, b, c so (i) v or x is of the form $a^i b^j$ (or $b^i c^j$) for some i, j such that $i + j < n$ (or) v or x is a string formed by repetition of only one symbol among a, b, c

When v or x is of the form $a^i b^j$, $v^2 = a^i b^j a^i b^j$ (or $x^2 = a^i b^j a^i b^j$). As v^2 is a substring of $u\,v^2\,w\,x^2\,y$ We cannot have $u\,v^2\,w\,x^2\,y$ of the form $a^m b^m c^m$ so $u\,v^2\,w\,x^2\,y \notin L$.

When both v and x are formed by repetition of a string symbol (eg $u = a^i$ and $v = b^j$ for some i and j, $i \leq n$, $j \leq n$) the string $u\,w\,y$ will contain the remaining symbol say a_1. Also a_1^n will be substring of way as a_1 does not occur in v or x. The number of occurrences of one of the other two symbols in uwy is less than n (recall $uvwxy = a^n b^n c^n$) and n is the number of occurrences of a_1. So $u\,v^0\,w\,x^0\,y = u\,w\,y \notin L$

Thus for any chance of v or x, we get a contradiction Therefore L is not context free.

— Problem 2 —

Show that $L = \{a^P \mid P \text{ is a prime}\}$ is not context - free language.

Solution

We use the following property of L. If $w \in L$, then $|w|$ is a prime.

Step 1: Suppose $L = L(G)$ is context - free. Let n be the natural number obtained by using the pumping lemma.

Step 2: Let P be a prime number greater than n. Then $z = a^P \in L$, we write $z = uvwxy$.

Step 3: By pumping lemma, $u v^0 w x^0 y = uwy \in L$.

So $|uwy|$ is a prime number. say q.

Let $|vx| = r$. Then $|u v^q w x^q y| = q + qr$.

As $q + q^r$ is not a prime, $u v^q w x^q y \notin L$. This is a contradiction. Therefore L is not context free.

4.12 Closure properties of CFL's

CFL's are closed under union, concatenation and star.

4.12.1 – Theorem

If L_1 and L_2 are CFL's, then $L_1 \cup L_2$, $L_1 . L_2$ and $L_1{}^*$ also denote the CFL.

Proof

Let L_1 and L_2 be two CFL's generated by the CFG's

$G_1 = (V_1, T_1, P_1, S_1)$ and

$G_2 = (V_2, T_2, P_2, S_2)$ respectively.

Assume that V_1 and V_2 are disjoint.

Case 1: Union of two CFL's is CFL. Let us consider the language L_3 generated by the grammar.

$G_3 = (V_1 \cup V_2 \cup S_3, T_1 \cup T_2, P_3, S_3)$ where

$S_3 = S_1 \mid S_2$

$P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 \mid S_2\}$

It is clear that grammar G_3 is context free and the language generated by this grammar is also context free.

It is easy to prove that $L_3 = L_1 \cup L_2$.

if we assume $w \in L_1$ then the possible derivation of w from S_3 is

$S_3 \Rightarrow S_1 \overset{*}{\Rightarrow} w$

if we assume $w \in L_2$ then the possible derivation of w from S_3 is

$S_3 \Rightarrow S_2 \overset{*}{\Rightarrow} w$

So if $w \in L_3$, one of the derivation

$S_3 \Rightarrow S_1$ or $S_3 \Rightarrow S_2$ is possible.

In the first case, all the variables in V_1 and all the terminals in T_1 may be used to get the derivation.

$S_1 \overset{*}{\Rightarrow} w$ which uses only the production in P_1

In the second case, all variables in V_2 and all the terminals in T_2 may be used to get the derivation $S_2 \overset{*}{\Rightarrow} w$ which uses only the production in P_2.

(i.e.,) $L_3 = L_1 \cup L_2$.

Thus it is proved that CFL's are closed under union.

Case 2: Concatenation of two CFL's is CFL.

Let us consider the language L_4 generated by the grammar.

$$G_4 = (V_1 \cup V_2 \cup S_4, T_1 \cup T_2, P_4 S_4)$$

where

$$S_4 \notin V_1 \cup V_2$$

$$P_4 = P_1 \cup P_2 \cup \{S_4 \rightarrow S_1 S_2\}$$

It is clear that the grammar G4 is context free and language generated by this grammar is context free (i.e.,)

$$L_4 = L_1 L_2$$

Thus it is proved that CFL's are closed under concatenation.

Case 3: CFL's are closed under star closure.

Let us consider the language generated by the grammar $G_5 = (V_1 \cup S_5, T_1, P_5, S_5)$ where

$$P_5 \rightarrow P_1 \cup \{S_5 \rightarrow S_1 S_5 \mid \Sigma\}$$

Hence the grammar G_5 is context free and the language generated by this grammar also content free.

$$L_5 = L_1{}^*$$

Thus it is proved that CFL's are closed under star closure.

4.5.1.2 Theorem

CFL's are not closed under complement.

If L is CFL, it is not true that complement of L is CFL.

Proof

Let us prove this theorem by contradiction Let L_1 and L_2 be CFL's, then $\overline{L_1}$ and $\overline{L_2}$ are also CFL's.

We have already proved that CFL's are closed under union, $\overline{L_1} \cup \overline{L_2}$ must be CFL.

Since we have assumed, that CFL's are closed under complementation $\overline{\overline{L_1} \cup \overline{L_2}}$ must be CFL.

According to de morgan's law

$$\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$$

So $L_1 \cap L_2$ must be CFL which is not true because CFL's are not closed under intersection.

Hence it is proved that CFL's are not closed under complementation.

4.12.3 Theorem

CFL's are not closed under Intersection.

If L_1 and L_2 are CFL's. It is not always true that $L_1 \cap L_2$ is CFL.

Proof

Let us prove this therem by taking counter examples consider the two content free languages.

$$L_1 = \{a^n b^n c^m \mid n \geq 0, m \geq 0\}$$
$$L_2 = \{a^n b^m c^m \mid n \geq 0, m \geq 0\}$$

Let us take

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ which is not a context free language.

Hence we can prove that that CFL's are not closed under intersection.

Short Answer Questions

1. What is an unit production?
2. Describe the general method of eliminating unit productions from the grammer.
3. Define CNF?
4. Define substitution?
5. Write the applications of CFG?
6. Define GNF?
7. Define useful and useless symbols in CFG?
8. What do you mean by reduction of grammer?
9. Write the steps to convert CFG to PDA.
10. What is an E-production?

Long Answer Questions

1. Identify nullable variables in the following productions and eliminate E - productions.

$$S \rightarrow aA \mid a \mid B \mid c$$
$$A \rightarrow aB \mid \epsilon$$
$$B \rightarrow aA$$
$$C \rightarrow cCD$$
$$D \rightarrow abd$$

2. Eliminate $\in$ productions from the following grammer.
$S \to ABAC$
$A \to aA \mid \in$
$B \to bB \mid \in$
$C \to c$
3. Simplify the following grammer.
$S \to aA \mid a \mid Bb \mid cC$
$A \to aB$
$B \to a \mid Aa$
$C \to cCD$
$D \to ddd$
4. Eliminate unit productions from the following grammer.
$S \to AB$
$A \to a$
$B \to C \mid b$
$C \to D$
$D \to E \mid bC$
$E \to d \mid Ab$
5. Eliminate unit productions from the following grammer.
$S \to A0 \mid B$
$B \to A \mid 11$
$A \to 0 \mid 12 \mid B$
6. Eliminate unit productions from the following grammer.
$S \to Aa \mid B \mid Ca$
$B \to aB \mid b$
$C \to Db \mid D$
$D \to E \mid d$
$E \to ab$
7. Eliminate $\in$ productions from the following grammer.
$S \to BAAB$
$A \to 0A2 \mid 2A0 \mid \in$
$B \to AB \mid 1B \mid \in$
8. State and prove pumping lemma for context free grammer?
9. Write the general procedure to convert CIG to GNF.
10. Convert the following CFG to CNF.
$S \to AaB \mid aaB$
$A \to \in$
$B \to bbA \mid \in$

11. Convert the following CFG to CNF.

$$S \to 0A \mid 1B$$
$$A \to 0AA \mid 1S \mid 1$$
$$B \to 1BB \mid 0S \mid 1$$

12. Convert the following grammer into GNF

$$S \to AB1 \mid 0$$
$$A \to 00A \mid B$$
$$B \to \mid A \mid$$

13. Convert the following grammer into GNF.

$$A \to BC$$
$$B \to CA \mid b$$
$$C \to AB \mid a$$

14. State and prove properties of context free grammer.
15. Obtain the PDA for the CFG given below

$$S \to aABB \mid aAA$$
$$A \to aBB \mid a$$
$$B \to bBB \mid A$$
$$C \to a$$

16. Obtain the PDA for the CFG given below

$$S \to aABC$$
$$A \to aB \mid a$$
$$B \to bA \mid b$$
$$C \to a$$

17. Write the steps to convert CFG to PDA.

Unit 5

TURING MACHINE

CHAPTER OUTLINE

5.1 · Turing Machine (TM)

Alan Turing introduced a new mathematical model called Turing Machine which is modified version of the PDA and is much more powerful than PDA.

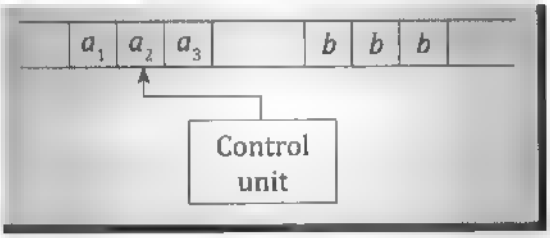
A PDA is not powerful enough to recognize context sensitive languages like $\{a^n, b^n, c^n: n \in N\}$, but Turing Machine is a generalized machine which can recognize all types of languages viz; regular languages (generaled from regular grammar) context free languages (generated from context free grammar), context sensitive languages (generated from context sensitive grammar and also the language generated from unrestricted grammar

It is a hypothetical machine, which can simulate any computer algorithm, no matter how complicated it is. A TM is a finite state machine with an infinite tape and a tape head that can read or write.

Model of Turing Machine

It is a finite automation with the following component.

- Tape
- Read write head
- Control unit.



Tape: Tape is used to store information and is divided into cells. Each cell can store only one symbol. The string to be scanned will be stored from the left position of the tape. The remaining infinity of cells each hold the blank (B), which is a special tape symbol that is not an input symbol. The tape is assumed to be infinite both on left and right side of the string.

Read write head: It can read/write a symbol from where it points. i.e., it scans one cell of the tape at a time. Initially it is at the leftmost cell that holds the input.

Control unit: The control unit is a finite set of states. It carries out the tasks based on transition table.

In one move the depending upon the symbol scanned by the tape head and the state of the finite control

- Changes state
- Replaces the symbol pointed by the tape head by another symbol and
- Moves its head left or right one cell tape cell and move left or right.

Definition

A Turing Machine (TM) is a seven tuple

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where,

Q – Set of finite states.

Σ – Set of input symbols. (Σ is a subset of Γ, excluding B)

Γ – Set of tape symbols

δ – Transitions from $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

$q_0 \in Q$ is a the start state of M.

B is a special symbol indicating blank character.

$F \subseteq Q$ is a set of final states.

Transitions: The transition function accepts two parameters namely a state, and a tape symbol and returns a new state after changing the tape symbol and position of read/write head.

The transition function has the form,

δ (state, tape symbol) = (next state, tape symbol, move towards Left/Right)

Example :

The transition $\delta(q_0, a) = (q_1, X, R)$ means that TM is in current state q_0, after scanning the symbol $a \in \Gamma$, TM enters into new state q_1, by replacing the tape symbol a by X and read/write head moves towards right.

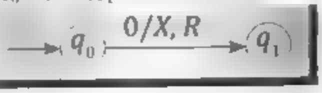
Transition Diagram for Turing Machine

The transition diagram consists of nodes corresponding to states of Turing Machine. An edge from state p to state q will have a label of the form $(X/Y, D)$ where x and y are tape symbols and D is the direction either `L' or `R' to denote `Left' or `Right' respectively.

Here X is the scanned symbol and Y is the symbol written on to the tape.

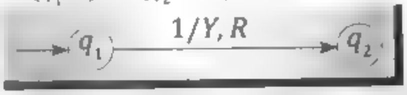
Example 1:

The graphical representation of $\delta(q_0, 0) = (q_1, X, R)$ is shown as



Example 2:

The graphical representation of $\delta(q_1, 1) = \delta(q_2, Y, R)$ is shown as



Transition table

The transition table for TM is a conventional tabular representation of a transition function

- The rows of the table corresponds to the states of TM
- The column corresponds to the tape symbols from Γ.
- The symbol B Indicates blank character. Usually the string ends with blank Character

δ					
Q/Γ	a	b	x	y	B
q_0	(q_1, x, R)	-	-	(q_2, y, R)	-
q_1	(q_1, a, R)	(q_2, y, L)	-	(q_2, y, R)	-
q_2	-	(q_2, a, L)	(q_0, x, R)	-	(q_2, B, R)

Note

The undefined entries in the table indicate that there are no- transitions defined i.e., dead state.

5.1.1 - Instantaneous Description of a TM (ID)

The current configuration of TM at any given instant can be described by an instantaneous description (ID). It describes.

(a) The name of the state in which the machine is presently in

(b) The symbols on the tape and

(c) The cell that is currently being scanned.

Definition: ID of a TM

An ID of a TM is a string $x\,q\,y$ where q is the current state, xy is the string made from tape symbols. The head points to the first character of the substring.

The initial ID is denoted by $q\,\alpha\,\beta$. Where q is the start state and tape head points to first symbol of α. The finial ID is denoted by $\alpha\,\beta\,q\,B$ where q is the final state and tape head points to B.

Example :

$$q\,x\,y \overset{*}{\vdash} x\,q\,y$$

if there is a transition $\delta(q, x) = (q, x, R)$

5.1.2 - Language Acceptance by of a TM L(M)

The language accepted by TM, denoted by L(M), is the set of those words in Σ^* that causes the TM to enter a final state, from state q_0 and the tape head positioned at the left most cell.

Definition: Language Accepted by a TM.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM.

The language $L(M)$ accepted by M is defined as

$$L(M) = \{w \mid q_0 w \overset{*}{\vdash} \alpha_1 p \alpha_2 \text{ where } w \in \Sigma^*, P \in F \text{ and } \alpha_1, \alpha_2 \in \Gamma\}^*$$

Where $q_1 w$ is the initial ID and $\alpha_1 P \alpha_2$ is the final ID. If set of all words $w \in \Sigma^*$ which causes M to move from start state q_t to final state P and halts, then we say that the string w is accepted by Turing Machine. The language accepted by TM is called recursively enumerable language.

Note

The Turing Machine can do one of the following things.

a. Halt and Accept by entering into final state

b. Halt and reject. This is possible if the transitions are not defined.

c. TM will never halt and enters into an infinite loop. There is no algorithm to determine when a given machine halts.

5.2 Construction of Turing Machine

Problem 1

Obtain a Turing Machine to accept the language L = $\{0^n, 1^n / n \geq 1\}$

Solution

General Procedure

It is given that the language accepted by TM should have n number of 0's followed by n number of 1's. Let us take for example 00001111.

Let q_0 be the start state of TM and the read – write head points to the first symbol of the string to be scanned.

$$\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \uparrow \\ q_0 \end{array}$$

Example :

1. Replace q_0 the left most 0 by X and change the state to q_1 and then move the read - write head towards right, because after a zero is replaced, we have to replace a corresponding 1 so that number of zeros matches with number of 1's.
2. Search the left most 1 and replace it by the symbol Y and move towards left.
3. Repeat step 1 and 2.

Design

Step 1: In state q_0, replace 0 by X, change the state to q_1 and move pointer towards right The transition can be of the form.

$\delta(q_0, 0) = (q_1, X, R)$

The resulting configuration is

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ & \uparrow & & & & & & \\ & q_1 & & & & & & \end{array}$$

Step 2: In state q_1, find left most 1 and replace it by Y and change the state to q_2. If we find any 0's or Y's while moving right, do not replace it with any other symbol.

(ie) $\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, 1) = (q_2, Y, L)$

The resulting configuration is

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & Y & 1 & 1 & 1 \\ & & & \uparrow & & & & \\ & & & q_2 & & & & \end{array}$$

Step 3: The read/write head has to move towards left to obtain left most zero. In order to identify left most zero, identify right most X. During this process. We may find Y's and 0's. Don't change its value and remain in the same state q_2.

$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_2, 0) = (q_2, 0, L)$

The resulting configuration is

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & Y & 1 & 1 & 1 \\ \uparrow & & & & & & & \\ q_2 & & & & & & & \end{array}$$

Step 4: To get leftmost 0, move the pointer to right without changing the symbol x with other symbol. But change the sate to q_0.

$\delta(q_2, X) = (q_0, X, R)$

The current configuration will be

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & Y & 1 & 1 & 1 \\ & \uparrow & & & & & & \\ & q_0 & & & & & & \end{array}$$

Step 5: Repeat step 1 to step 4 to get the configuration shown below

$$\begin{array}{cccccccc} X & X & X & X & Y & Y & Y & Y \\ & & & & \uparrow & & & \\ & & & & q_0 & & & \end{array}$$

Step 6: In the state q_0, if the scanned symbol is Y, it means there are no more 0's. To check there are no more l's move the pointer towards right by changing the state to q_3

$\delta(q_0, Y) = (q_3, Y, R)$

Step 7: In state q_3, we should see there are only Y's and no more l's.

$\delta(q_3, Y) = (q_3, Y, R)$

Repeatedly applying this transition, the following configuration is obtained.

$$\begin{array}{ccccccccc} X & X & X & X & Y & Y & Y & Y & B \\ & & & & & & & & \uparrow \\ & & & & & & & & q_3 \end{array}$$

Step 8: Since the string ends with infinite number of blanks, change the state to q_4 which is a final state.

$\delta (q_3, B) = (q_4, B, R)$

∴ The Turing Machine to accept the language $L = \{a^n b^n \mid n \geq 1\}$ is defined by

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0 , 1\}$

$\Gamma = \{0, 1, X, Y, B\}$

$q_0 \in Q$ is the start state of machine

$B \in \Gamma$ is the blank symbol.

$F = \{q_4\}$ is the final state

δ: is shown below

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, 1) = (q_2, Y, L)$

$\delta(q_2, y) = (q_2, Y, L)$

$\delta(q_2, 0) = (q_2, 0, L)$

$\delta(q_2, X) = (q_0, X, R)$

$\delta(q_0, Y) = (q_3, Y, R)$

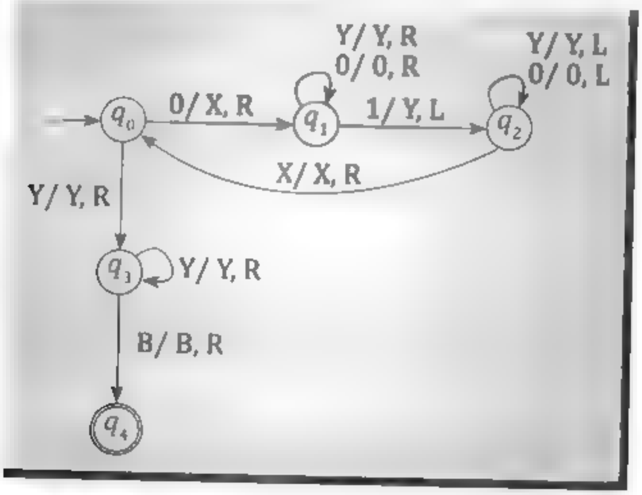
$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, B) = (q_4, B, R)$

The transitions can also be represented by transition table.

δ					
Q/Γ	0	b	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1, 0, R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, X, R)	$(q2, Y, L)$	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Turing Machine can also be represented by transition diagram.



To accept the string

The sequence of moves made by the TM to check whether the string 0011 is accepted or not is.

$q_0 0011$ (initial ID) $\vdash$ $Xq_1 011$ $\vdash$ $X0q_1 11$ $\vdash$ $Xq_2 0Y1$

$\vdash$ $q_2 X0Y1$ $\vdash$ $Xq_0 0Y1$ $\vdash$ $XXq_1 Y1$

$\vdash$ $XXYq_1 1$ $\vdash$ $XXq_2 YY$ $\vdash$ $Xq_2 XYY$

$\vdash$ $XXq_0 YY$ $\vdash$ $XXYq_3 Y$ $\vdash$ $XXYYq_3$

$\vdash$ $XXYYBq_4$
(Final ID)

Since the final state q_4 is reached, the string 0011 is accepted.

PROBLEM 2

Show that L = $\{a^P \mid P$ is a prime$\}$ is not context - free language.

SOLUTION

We use the following property of L. If $w \in L$, then $|w|$ is a prime.

Step 1: Suppose L = L(G) is context - free. Let n be the natural number obtained by using th pumping lemma.

Step 2: Let P be a prime number greater than n. Then $z = a^P \in L$, we write $z = u\,v\,w\,x\,y$.

Step 3: By pumping lemma, $u\,v^0\,w\,x^0\,y = u\,w\,y \in L$.

So $|u\,w\,y|$ is a prime number. say q.

Let $|vx| = r$. Then $|u\,v^q\,w\,x^q\,y| = q + qr$.

As $q + q^r$ is not a prime, $u\,v^q\,w\,x^q\,y \notin L$. This is a contradiction. Therefore L is not context free.

4.12 CLOSURE PROPERTIES OF CFL's

CFL's are closed under union, concatenation and star.

4.12.1 Theorem

If L_1 and L_2 are CFL's, then $L_1 \cup L_2$, $L_1 . L_2$ and $L_1{}^*$ also denote the CFL.

Proof

Let L_1 and L_2 be two CFL's generated by the CFG's

$G_1 = (V_1, T_1, P_1, S_1)$ and

$G_2 = (V_2, T_2, P_2, S_2)$ respectively.

Assume that V_1 and V_2 are disjoint.

Case 1: Union of two CFL's is CFL. Let us consider the language L_3 generated by the grammar.

$G_3 = (V_1 \cup V_2 \cup S_3, T_1 \cup T_2, P_3, S_3)$ where

$S_3 = S_1 \mid S_2$

$P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 \mid S_2\}$

It is clear that grammar G_3 is context free and the language generated by this grammar is also context free.

It is easy to prove that $L_3 = L_1 \cup L_2$.

If we assume $w \in L_1$ then the possible derivation of w from S_3 is

$S_3 \Rightarrow S_1 \overset{*}{\Rightarrow} w$

If we assume $w \in L_2$ then the possible derivation of w from S_3 is

$S_3 \Rightarrow S_2 \overset{*}{\Rightarrow} w$

So if $w \in L_3$, one of the derivation

$S_3 \Rightarrow S_1$ or $S_3 \Rightarrow S_2$ is possible.

In the first case, all the variables in V_1 and all the terminals in T_1 may be used to get the derivation.

$S_1 \overset{*}{\Rightarrow} w$ which uses only the production in P_1

In the second case, all variables in V_2 and all the terminals in T_2 may be used to get the derivation $S_2 \overset{*}{\Rightarrow} w$ which uses only the production in P_2.

(i.e.,) $L_3 = L_1 \cup L_2$.

Thus it is proved that CFL's are closed under union.

Case 2: Concatenation of two CFL's is CFL.

Let us consider the language L_4 generated by the grammar.

$G_4 = (V_1 \cup V_2 \cup S_4, T_1 \cup T_2, P_4 S_4)$

where

$S_4 \notin V_1 \cup V_2$

$P_4 = P_1 \cup P_2 \cup \{S_4 \rightarrow S_1 S_2\}$

It is clear that the grammar G4 is context free and language generated by this grammar is context free (i.e.,)

$L_4 = L_1 L_2$

Thus it is proved that CFL's are closed under concatenation.

Case 3: CFL's are closed under star closure.

Let us consider the language generated by the grammar $G_5 = (V_1 \cup S_5, T_1, P_5, S_5)$

where

$P_5 \rightarrow P_1 \cup \{S_5 \rightarrow S_1 S_5 \mid \Sigma\}$

Hence the grammar G_5 is context free and the language generated by this grammar also content free.

$L_5 = L_1^*$

Thus it is proved that CFL's are closed under star closure.

4.12.2 Theorem

CFL's are not closed under complement.

If L is CFL, it is not true that complement of L is CFL.

Proof

Let us prove this theorem by contradiction Let L_1 and L_2 be CFL's, then $\bar{L}_1$ and $\bar{L}_2$ are also CFL's.

We have already proved that CFL's are closed under union, $\bar{L}_1 \cup \bar{L}_2$ must be CFL.

Since we have assumed, that CFL's are closed under complementation $\overline{\bar{L}_1 \cup \bar{L}_2}$ must be CFL.

According to de morgan's law

$$\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$$

So $L_1 \cap L_2$ must be CFL which is not true because CFL's are not closed under intersection.

Hence it is proved that CFL's are not closed under complementation.

4.12.3 Theorem

CFL's are not closed under Intersection.

If L_1 and L_2 are CFL's. It is not always true that $L_1 \cap L_2$ is CFL.

Proof

Let us prove this therem by taking counter examples consider the two content free languages.

$$L_1 = \{a^n b^n c^m \mid n \geq 0, m \geq 0\}$$
$$L_2 = \{a^n b^m c^m \mid n \geq 0, m \geq 0\}$$

Let us take

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ which is not a context free language.

Hence we can prove that that CFL's are not closed under intersection.

Short Answer Questions

1. What is an unit production?
2. Describe the general method of eliminating unit productions from the grammer.
3. Define CNF?
4. Define substitution?
5. Write the applications of CFG?
6. Define GNF?
7. Define useful and useless symbols in CFG?
8. What do you mean by reduction of grammer?
9. Write the steps to convert CFG to PDA.
10. What is an E-production?

Long Answer Questions

1. Identify nullable variables in the following productions and eliminate E - productions.

$$S \rightarrow aA \mid a \mid B \mid c$$
$$A \rightarrow aB \mid \in$$
$$B \rightarrow aA$$
$$C \rightarrow cCD$$
$$D \rightarrow abd$$

2. Eliminate $\in$ productions from the following grammer.

$$S \to ABAC$$
$$A \to aA \mid \in$$
$$B \to bB \mid \in$$
$$C \to c$$

3. Simplify the following grammer.

$$S \to aA \mid a \mid Bb \mid cC$$
$$A \to aB$$
$$B \to a \mid Aa$$
$$C \to cCD$$
$$D \to ddd$$

4. Eliminate unit productions from the following grammer.

$$S \to AB$$
$$A \to a$$
$$B \to C \mid b$$
$$C \to D$$
$$D \to E \mid bC$$
$$E \to d \mid Ab$$

5. Eliminate unit productions from the following grammer.

$$S \to A0 \mid B$$
$$B \to A \mid 11$$
$$A \to 0 \mid 12 \mid B$$

6. Eliminate unit productions from the following grammer.

$$S \to Aa \mid B \mid Ca$$
$$B \to aB \mid b$$
$$C \to Db \mid D$$
$$D \to E \mid d$$
$$E \to ab$$

7. Eliminate $\in$ productions from the following grammer.

$$S \to BAAB$$
$$A \to 0A2 \mid 2A0 \mid \in$$
$$B \to AB \mid 1B \mid \in$$

8. State and prove pumping lemma for context free grammer?
9. Write the general procedure to convert CIG to GNF.
10. Convert the following CFG to CNF.

$$S \to AaB \mid aaB$$
$$A \to \in$$
$$B \to bbA \mid \in$$

11. Convert the following CFG to CNF.

$$S \rightarrow 0A \mid 1B$$
$$A \rightarrow 0AA \mid 1S \mid 1$$
$$B \rightarrow 1BB \mid 0S \mid 1$$

12. Convert the following grammer into GNF

$$S \rightarrow AB1 \mid 0$$
$$A \rightarrow 00A \mid B$$
$$B \rightarrow \mid A \mid$$

13. Convert the following grammer into GNF.

$$A \rightarrow BC$$
$$B \rightarrow CA \mid b$$
$$C \rightarrow AB \mid a$$

14. State and prove properties of context free grammer.
15. Obtain the PDA for the CFG given below

$$S \rightarrow aABB \mid aAA$$
$$A \rightarrow aBB \mid a$$
$$B \rightarrow bBB \mid A$$
$$C \rightarrow a$$

16. Obtain the PDA for the CFG given below

$$S \rightarrow aABC$$
$$A \rightarrow aB \mid a$$
$$B \rightarrow bA \mid b$$
$$C \rightarrow a$$

17. Write the steps to convert CFG to PDA.

●★●★●

Unit 5

TURING MACHINE

CHAPTER OUTLINE

5.1 - Turing Machine (TM)

Alan Turing introduced a new mathematical model called Turing Machine which is modified version of the PDA and is much more powerful than PDA.

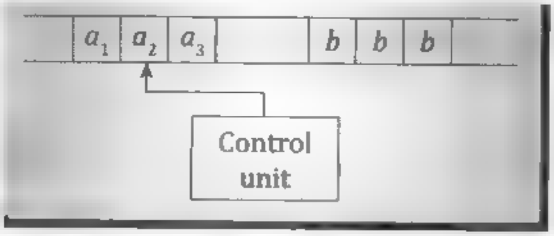
A PDA is not powerful enough to recognize context sensitive languages like $\{a^n, b^n, c^n: n \in N\}$, but Turing Machine is a generalized machine which can recognize all types of languages viz; regular languages (generaled from regular grammar) context free languages (generated from context free grammar), context sensitive languages (generated from context sensitive grammar and also the language generated from unrestricted grammar

It is a hypothetical machine, which can simulate any computer algorithm, no matter how complicated it is. A TM is a finite state machine with an infinite tape and a tape head that can read or write.

Model of Turing Machine

It is a finite automation with the following component.

- Tape
- Read write head
- Control unit.



Tape: Tape is used to store information and is divided into cells. Each cell can store only one symbol. The string to be scanned will be stored from the left position of the tape. The remaining infinity of cells each hold the blank (B), which is a special tape symbol that is not an input symbol. The tape is assumed to be infinite both on left and right side of the string.

Read write head: It can read/write a symbol from where it points. i.e., it scans one cell of the tape at a time. Initially it is at the leftmost cell that holds the input.

Control unit: The control unit is a finite set of states. It carries out the tasks based on transition table.

In one move the depending upon the symbol scanned by the tape head and the state of the finite control

- Changes state
- Replaces the symbol pointed by the tape head by another symbol and
- Moves its head left or right one cell tape cell and move left or right.

Definition

A Turing Machine (TM) is a seven tuple

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where, Q – Set of finite states.

Σ – Set of input symbols. (Σ is a subset of Γ, excluding B)

Γ – Set of tape symbols

δ – Transitions from $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

$q_0 \in Q$ is a the start state of M.

B is a special symbol indicating blank character.

$F \subseteq Q$ is a set of final states.

Transitions: The transition function accepts two parameters namely a state, and a tape symbol and returns a new state after changing the tape symbol and position of read/write head.

The transition function has the form,

δ (state, tape symbol) = (next state, tape symbol, move towards Left/Right)

Example :

The transition $\delta(q_0, a) = (q_1, X, R)$ means that TM is in current state q_0, after scanning the symbol $a \in \Gamma$, TM enters into new state q_1, by replacing the tape symbol a by X and read/write head moves towards right.

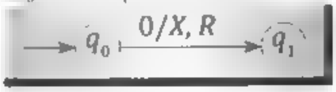
Transition Diagram for Turing Machine

The transition diagram consists of nodes corresponding to states of Turing Machine. An edge from state p to state q will have a label of the form $(X/Y, D)$ where x and y are tape symbols and D is the direction either `L' or `R' to denote `Left' or `Right' respectively.

Here X is the scanned symbol and Y is the symbol written on to the tape.

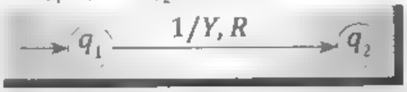
Example 1:

The graphical representation of $\delta(q_0, 0) = (q_1, X, R)$ is shown as



Example 2:

The graphical representation of $\delta(q_1, 1) = \delta(q_2, Y, R)$ is shown as



Transition table

The transition table for TM is a conventional tabular representation of a transition function.

- The rows of the table corresponds to the states of TM
- The column corresponds to the tape symbols from Γ.
- The symbol B Indicates blank character. Usually the string ends with blank Character.

δ					
Q/Γ	*a*	*b*	*x*	*y*	*B*
q_0	(q_1, x, R)	-	-	(q_2, y, R)	-
q_1	(q_1, a, R)	(q_2, y, L)	-	(q_2, y, R)	-
q_2	-	(q_2, a, L)	(q_0, x, R)	-	(q_2, B, R)

Note

The undefined entries in the table indicate that there are no- transitions defined i.e., dead state.

5.1.1 – Instantaneous Description of a TM (ID)

The current configuration of TM at any given instant can be described by an instantaneous description (ID). It describes.

(a) The name of the state in which the machine is presently in

(b) The symbols on the tape and

(c) The cell that is currently being scanned.

Definition: ID of a TM

An ID of a TM is a string *x q y* where q is the current state, *xy* is the string made from tape symbols. The head points to the first character of the substring.

The initial ID is denoted by $q\,\alpha\,\beta$ Where q is the start state and tape head points to first symbol of α. The final ID is denoted by $\alpha\,\beta\,q\,B$ where q is the final state and tape head points to B.

Example:

$$q\,x\,y \vdash^{*} x\,q\,y$$

if there is a transition $\delta(q, x) = (q, x, \text{R})$

5.1.2 – Language Acceptance by of a TM L(M)

The language accepted by TM, denoted by L(M), is the set of those words in Σ* that causes the TM to enter a final state, from state q_0 and the tape head positioned at the left most cell.

Definition: Language Accepted by a TM.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM.

The language $L(M)$ accepted by M is defined as

$$L(M) = \{w \mid q_0 w \overset{*}{\vdash} \alpha, p\, \alpha_2 \text{ where } w \in \Sigma^*, P \in F \text{ and } \alpha_1, \alpha_2 \in \Gamma\}^*$$

Where $q_0 w$ is the initial ID and $\alpha_1 P \alpha_2$ is the final ID. If set of all words $w \in \Sigma^*$ which causes M to move from start state q_0 to final state P and halts, then we say that the string w is accepted by Turing Machine. The language accepted by TM is called recursively enumerable language.

Note

The Turing Machine can do one of the following things.

a. Halt and Accept by entering into final state

b. Halt and reject. This is possible if the transitions are not defined.

c. TM will never halt and enters into an infinite loop. There is no algorithm to determine when a given machine halts.

5.2 Construction of Turing Machine

Problem 1

Obtain a Turing Machine to accept the language $L = \{0^n, 1^n / n \geq 1\}$

Solution

General Procedure

It is given that the language accepted by TM should have n number of 0's followed by n number of 1's. Let us take for example 00001111.

Let q_0 be the start state of TM and the read – write head points to the first symbol of the string to be scanned.

```
0   0   0   0   1   1   1   1
↑
q0
```

Example :

1. Replace q_0 the left most 0 by X and change the state to q_1 and then move the read – write head towards right, because after a zero is replaced, we have to replace a corresponding 1 so that number of zeros matches with number of 1's.
2. Search the left most 1 and replace it by the symbol Y and move towards left.
3. Repeat step 1 and 2.

Design

Step 1: In state q_0, replace 0 by X, change the state to q_1 and move pointer towards right. The transition can be of the form.

$\delta(q_0, 0) = (q_1, X, R)$

The resulting configuration is

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ & \uparrow & & & & & & \\ & q_1 & & & & & & \end{array}$$

Step 2: In state q_1, find left most 1 and replace it by Y and change the state to q_2. If we find any 0's or Y's while moving right, do not replace it with any other symbol.

(ie) $\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, 1) = (q_2, Y, L)$

The resulting configuration is

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & Y & 1 & 1 & 1 \\ & & & \uparrow & & & & \\ & & & q_2 & & & & \end{array}$$

Step 3: The read/write head has to move towards left to obtain left most zero. In order to identify left most zero, identify right most X. During this process. We may find Y's and 0's. Don't change its value and remain in the same state q_2.

$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_2, 0) = (q_2, 0, L)$

The resulting configuration is

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & Y & 1 & 1 & 1 \\ \uparrow & & & & & & & \\ q_2 & & & & & & & \end{array}$$

Step 4: To get leftmost 0, move the pointer to right without changing the symbol x with other symbol. But change the sate to q_0.

$\delta(q_2, X) = (q_0, X, R)$

The current configuration will be

$$\begin{array}{cccccccc} X & 0 & 0 & 0 & Y & 1 & 1 & 1 \\ & \uparrow & & & & & & \\ & q_0 & & & & & & \end{array}$$

tep 5: Repeat step 1 to step 4 to get the configuration shown below

$$\begin{array}{cccccccc} X & X & X & X & Y & Y & Y & Y \\ & & & & \uparrow & & & \\ & & & & q_0 & & & \end{array}$$

Step 6: In the state q_0, if the scanned symbol is Y, it means there are no more 0's. To check there are no more 1's move the pointer towards right by changing the state to q_3

$\delta(q_0, Y) = (q_3, Y, R)$

Step 7: In state q_3, we should see there are only Y's and no more 1's.

$\delta(q_3, Y) = (q_3, Y, R)$

Repeatedly applying this transition, the following configuration is obtained.

$$\begin{array}{ccccccccc} X & X & X & X & Y & Y & Y & Y & B \\ & & & & & & & & \uparrow \\ & & & & & & & & q_3 \end{array}$$

Step 8: Since the string ends with infinite number of blanks, change the state to q_4 which is a final state.

$\delta(q_3, B) = (q_4, B, R)$

The Turing Machine to accept the language $L = \{a^n b^n \mid n \geq 1\}$ is defined by

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, X, Y, B\}$

$q_0 \in Q$ is the start state of machine

$B \in \Gamma$ is the blank symbol.

$F = \{q_4\}$ is the final state

δ: is shown below

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, 1) = (q_2, Y, L)$

$\delta(q_2, y) = (q_2, Y, L)$

$\delta(q_2, 0) = (q_2, 0, L)$

$\delta(q_2, X) = (q_0, X, R)$

$\delta(q_0, Y) = (q_3, Y, R)$

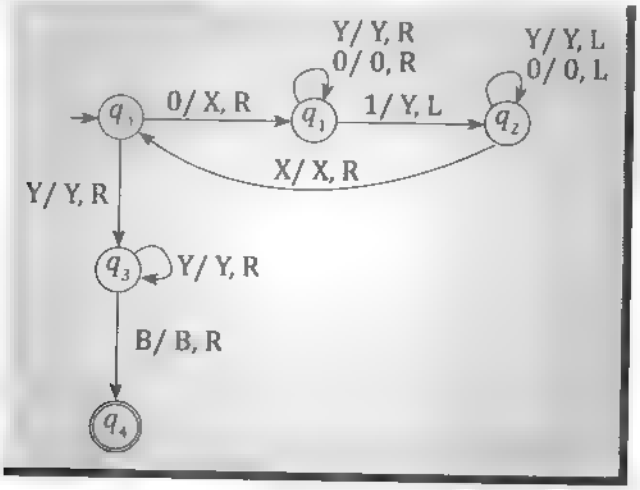
$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, B) = (q_4, B, R)$

The transitions can also be represented by transition table.

δ					
Q/Γ	0	b	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1, 0, R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, X, R)	$(q2, Y, L)$	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Turing Machine can also be represented by transition diagram.



To accept the string

The sequence of moves made by the TM to check whether the string 0011 is accepted or not is.

$q_0 0011$ ⊢ $Xq_1 011$ ⊢ $X0q_1 11$ ⊢ $Xq_2 0Y1$
(initial ID)

⊢ $q_2 X0Y1$ ⊢ $Xq_0 0Y1$ ⊢ $XXq_1 Y1$
⊢ $XXYq_1 1$ ⊢ $XXq_2 YY$ ⊢ $Xq_2 XYY$
⊢ $XXq_0 YY$ ⊢ $XXYq_3 Y$ ⊢ $XXYYq_3$
⊢ $XXYYBq_4$
(Final ID)

Since the final state q_4 is reached, the string 0011 is accepted.

Problem 2

Construct a Turing Machine to accept the language

$L(M) = \{0^n 1^n 2^n \mid n \geq 1\}$

Solution

It is given that the language accepted by TM should have n number of 0's followed by n number of 1's followed by n number of 2's

Let q_0 be the start state of TM and the read write head points to the first symbol of the string to be scanned.

Example :

```
0   0   0   0   1   1   1   1   2   2   2   2
↑
q0
```

Replace the left most 0 by X, 1 by Y and 2 by Z so that the number of 0's matches with number of 1's and 2's

General Procedure

Step 1: In state q_0 replace 0 by X and change the state to q_1, move the pointer towards right. The transition can be of the form.

$\delta(q_0, 0) = (q_1, X, R)$

The current configuration is

```
X   0   0   0   1   1   1   1   2   2   2   2
    ↑
    q1
```

Step 2: In the state q_1 find leftmost 1 and replace it by Y and change the state to q_2. Move the pointer towards right.

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, 1) = (q_2, Y, R)$

And the configuration will be,

```
X   0   0   0   Y   1   1   1   2   2   2   2
                    ↑
                    q2
```

Step 3: In state q_2, find left most 2 and replace it with Z by changing the state to q_3 and pointer towards left. When we are searching for the symbol 2, if we encounter 1 or Z, don't change the symbol and remain in state q_2.

$\delta(q_2, 1) = (q_2, 1, R)$

$\delta(q_2, Z) = (q_2, Z, R)$

$\delta(q_2, 2) = (q_3, Z, L)$

Step 4: Search for the rightmost X to get leftmost 0. During this process, if the symbols such as Z's, 1's, Y's, 0's are scanned, do not change the symbol and state but simply move the pointer to the left.

$\delta(q_3, Z) = (q_3, Z, L)$

$\delta(q_3, 1) = (q_3, 1, L)$

$\delta(q_3, Y) = (q_3, Y, L)$

$\delta(q_3, 0) = (q_3, 0, L)$

On encountering X, change the state to q_0, and move the pointer towards right to get leftmost 0.

$\delta(q_3, X) = (q_0, X, R)$

The configuration will be,

```
X   0   0   0   Y   1   1   1   Z   2   2   2
    ↑
    q0
```

Step 5: Repeat step 1 to step 4 till the following configuration is obtained

```
X   X   X   X   Y   Y   Y   Y   Z   Z   Z   Z
                    ↑
                    q0
```

Step 6: In the state q_0, if the scanned symbol is Y, it means there are no more 0's . to check there are no more 1's move the pointer towards right by changing the state as q_4

$\delta(q_0, Y) = (q_4, Y, R)$

Step 7: In state q_4, we should see there are only Y's and no more 1's by moving the pointer towards right as long as the scanned symbol is Y.

$\delta(q_4, Y) = (q_4, Y, R)$

The current configuration will be

```
X   X   X   X   Y   Y   Y   Y   Z   Z   Z   Z
                                ↑
                                q4
```

Step 8: In state q_4, if we encounter Z, it means there are no I's on scanning the first Z, change the state to q_5.

$\delta(q_4, Z) = (q_5, Z, R)$

Step 9: In state q_5, we should see there are only Z's and no more 2's by moving the pointer towards right as long as the scanned symbol is Z

$\delta(q_5, Z) = (q_5, Z, R)$

The Current configuration will be

$$\begin{array}{ccccccccccccc} X & X & X & X & Y & Y & Y & Y & Z & Z & Z & Z & B \\ &&&&&&&&&&&& \uparrow \\ &&&&&&&&&&&& q_5 \end{array}$$

Step 10: Since the string ends with infinite number of blanks, change the state to q_6 which is a final state.

$\delta(q_5, B) = (q_6, B, R)$

. The Turing Machine to accept the language

$L = \{0^n 1^n 2^n \mid n > 1\}$ is defined by

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$\Sigma = \{0 , 1\}$

$\Gamma = \{0, 1, 2, X, Y, Z, B\}$

$q_0 \in Q$ is the start state of machine

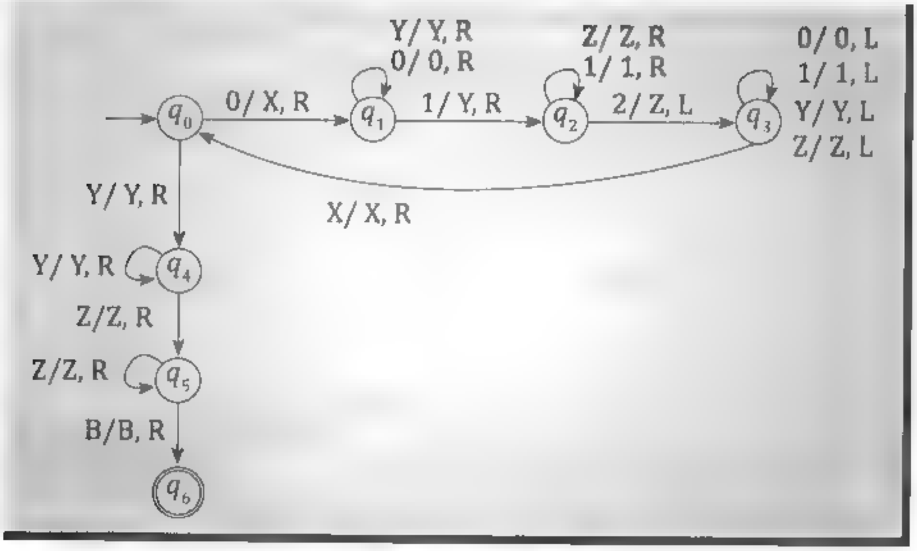
$B \in \Gamma$ is the blank symbol.

$F = \{q_6\}$ is the final state.

δ is shown below using transition table

δ							
Γ / Q	0	1	2	X	Y	Z	B
$\rightarrow q_0$	(q_1, X, R)	-	-	-	(q_4, Y, R)	-	-
q_1	$(q_1, 0, R)$	(q_2, Y, R)	-	-	(q_1, Y, R)	-	-
q_2	-	$(q_2, 1, R)$	(q_3, Z, L)	-	-	(q_2, Z, R)	-
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	-	(q_0, X, R)	(q_3, Y, L)	(q_3, Z, L)	-
q_4	-	-	-	-	(q_4, Y, R)	(q_5, Z, R)	-
q_5	-	-	-	-	-	(q_5, Z, R)	(q_6, B, R)
* $q6$	-	-	-	-	-	-	-

The transition diagram



PROBLEM 3

Obtain a TM to accept a string w of a's and b's such that $N_a(w)$ is equal to $N_b(w)$ is equal to $N_b(w)$.

SOLUTION

It is given that the language accepted by TM can have a's and b's at any positions provided number of a's should be equal to number of b's. For example *abbaba*.

General Procedure

Let q_0 be the start state and the read write head points to the first symbol of the string to be scanned. The general procedure to design a TM will result in three cases depending on the input symbol to be scanned on the state q_0.

1. On encountering a 2. On encountering b 3. On encountering B

Case 1: On encountering a

In state q_0, if we encounter a replace it by X. Then skip all subsequent symbols till we read b and replace b by Y. Now repeat any one of the three cases based on the next symbol to be scanned.

Consider the situation

$$\begin{array}{ccccccccc} X & X & a & Y & a & Y & b & b & B \\ & & \uparrow & & & & & & \\ & & q_0 & & & & & & \end{array}$$

Step 1: In state q_0, on reading a, change the state to q_1, replace a by X and move the pointer towards right. The resulting transition is

$\delta(q_0, a) = (q_1, X, R)$

The resulting configuration is shown below

$$\begin{array}{ccccccccc} X & X & X & Y & a & Y & b & b & B \\ & & \uparrow & & & & & & \\ & & q_1 & & & & & & \end{array}$$

Step 2: In state q_1, move the pointer towards right in search of b and replace it with Y. The transitions are.

$\delta(q_1, a) = (q_1, a, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, b) = (q_2, Y, L)$

The resulting configuration is shown below

$$\begin{array}{ccccccccc} X & X & X & Y & a & Y & Y & b & B \\ & & & & & \uparrow & & & \\ & & & & & q_2 & & & \end{array}$$

Step 3: In state q_2, move the pointer towards left. On encountering X, change the state to q_0

$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_2, a) = (q_2, a, L)$

$\delta(q_2, X) = (q_0, X, R)$

The resulting configuration is

$$\begin{array}{ccccccccc} X & X & X & Y & a & Y & Y & b & B \\ & & & \uparrow & & & & & \\ & & & q_0 & & & & & \end{array}$$

Step 4: In the state q_0, on reading Y indicates so far the number of a's is equal to number of b's so move the pointer towards right.

$\delta(q_0, Y) = (q_0, Y, R)$

The resulting configuration is

$$\begin{array}{ccccccccc} X & X & X & Y & a & Y & Y & b & B \\ & & & & \uparrow & & & & \\ & & & & q_0 & & & & \end{array}$$

Repeat one of the three case.

Case 2: One encounters *b* in state q_0, if we find b replace it with X. Then skip all subsequent symbols till we read a and replace a by Y. Now repeat any one of the three cases based on the next symbol to be scanned.

Consider the situation

$$\begin{array}{cccccccc} X & X & b & Y & b & a & a & B \\ & & \uparrow & & & & & \\ & & q_0 & & & & & \end{array}$$

Step 1: In state q_0, on reading b, change the state to q_3, replace b by X and move the pointer towards rights to get left most a.

$\delta(q_0, b) = (q_3, X, R)$

The resulting configuration is

$$\begin{array}{ccccccccc} X & X & X & Y & b & Y & a & a & B \\ & & & \uparrow & & & & & \\ & & & q_3 & & & & & \end{array}$$

Step 2: In state q_3, move the pointer towards right to find a and replace it with Y.

$\delta(q_3, b) = (q_3, b, R)$

$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, a) = (q_4, Y, L)$

The resulting configuration is

$$\begin{array}{ccccccccc} X & X & X & Y & b & Y & Y & a & B \\ & & & & & \uparrow & & & \\ & & & & & q_4 & & & \end{array}$$

Step 3: In state q_4, move the pointer towards left on encountering X, change the state to q_0 and move the pointer towards right.

$\delta(q_4, Y) = (q_4, Y, L)$

$\delta(q_4, b) = (q_4, b, L)$

$\delta(q_4, X) = (q_0, X, R)$

The resulting configuration is

$$\begin{array}{ccccccccc} X & X & X & Y & b & Y & Y & a & B \\ & & & \uparrow & & & & & \\ & & & q_0 & & & & & \end{array}$$

Step 4: In state q_0, on reading Y indicates so far the number of a's are equal to number of b's so move the pointer towards right.

$$\delta(q_0, Y) = (q_0, Y, R)$$

The resulting configuration is

$$\begin{array}{ccccccccc} X & X & X & Y & b & Y & Y & a & B \\ & & & & \uparrow & & & & \\ & & & & q_0 & & & & \end{array}$$

Repeat one of the three cases

Case 3: One encounters B in state q_0, if we encounter B, move the pointer towards right and change the state to q_f which is the final state.

$$\delta(q_0, B) = (q_f, B, R)$$

So, the TM to accept strings of a's and b's such that number of a's is equal to number of b's is defined as

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

where,

$$\Sigma = (a, b)$$

$$\Gamma = (a, b, X, Y, B)$$

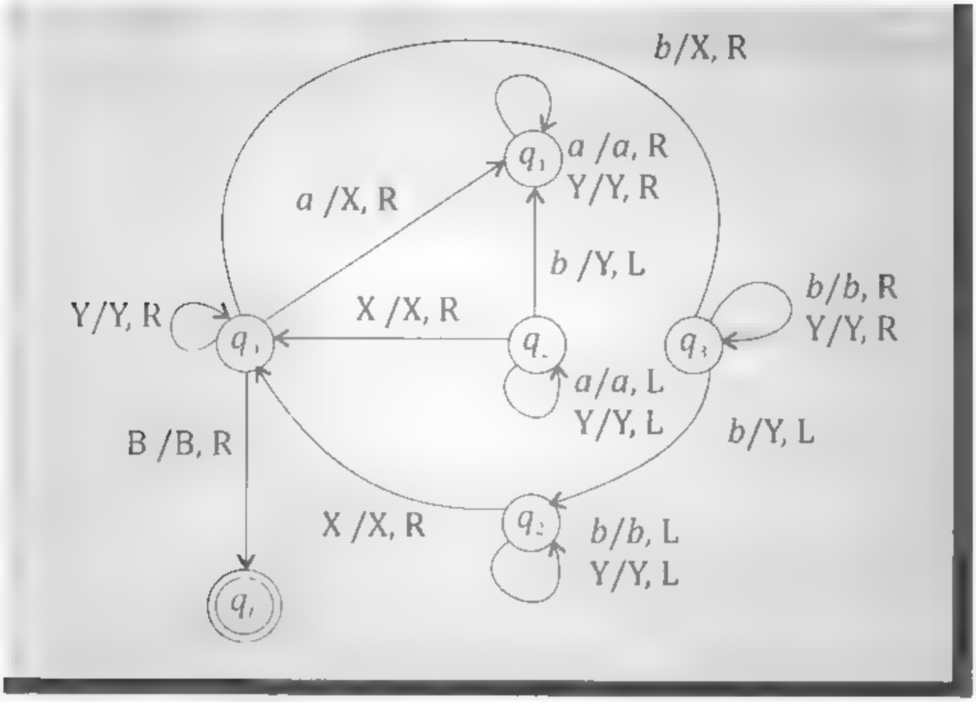
q_0 is the start state

B is the Blank character

$$F = (q_f)$$

δ = is shown below using the transition table

δ					
Q/Γ	a	b	X	Y	B
$\rightarrow q_0$	q_1, X, R	q_3, X, R	-	q_0, Y, R	q_f, B, R
q_1	q_1, a, R	q_2, Y, L	-	q_1, Y, R	
q_2	q_2, a, L		q_0, X, R	q_2, Y, L	
q_3	q_4, Y, L	q_3, b, R	-	q_3, Y, R	
q_4	-	q_4, b, L	q_0, X, R	q_4, Y, L	
$* q_f$	-	-	-	-	



5.3 - Types of Turing Machines

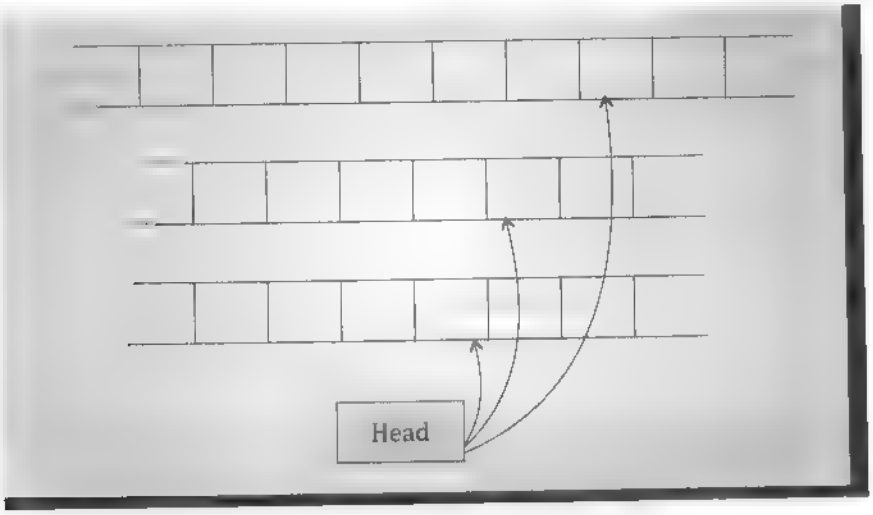
There are a number of other types of turing machines in addition to the one which are have already seen, such as turing machines with multiple tapes, turing machine having one tape but with multiple heads, turing machine with two dimensional tapes, non deterministic turing machines etc. All these turing machines are equally powerful. That is, what one type of turing machine computes any other can also compute the same. However, the efficiency of computation, that is, how fast they can compute may vary.

1. **Multi tape TM**

   Multi tape Turing machines have multiple tapes where each tape is accented with a separate head. It consists of a finite control with K - tape heads and K - tapes. Each tape is infinite in both directions. Each head can move independently of the other heads. Initially the input is on tapel and others are blank At first the first tape is occupied by the input and the other tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads. On a single move depending upon the state of the finite control and the symbol scanned by each of the tape heads the machine can

   - Change the state
   - Print a new symbol on each of the cells scanned by its tape heads.
   - Move each of its tape heads independently one cell to the left of right.

The language accepted by a one-way infinite tape TM is also accepted by a multi tape TM.



2. **Non deterministic TM**

   A non deterministic TM is a generalization of the standard TM for which every configuration may yield none, or one or more than one next configurations. In contrast to the deterministic turing machines, for which a computation is a sequence of configurations, i.e., a computation of a non deterministic TM is a tree of configurations that can be reached from the start configuration. For a given state and the tape symbol scanned by the tape head, the machine has a finite number of choices for the next move. Each choice consists of a new state, a tape symbol to print and a direction of head motion. If the input is accepted at least by one path among all the different possible paths, then the input is accepted by the TM.

3. **Multi - dimensional TM**

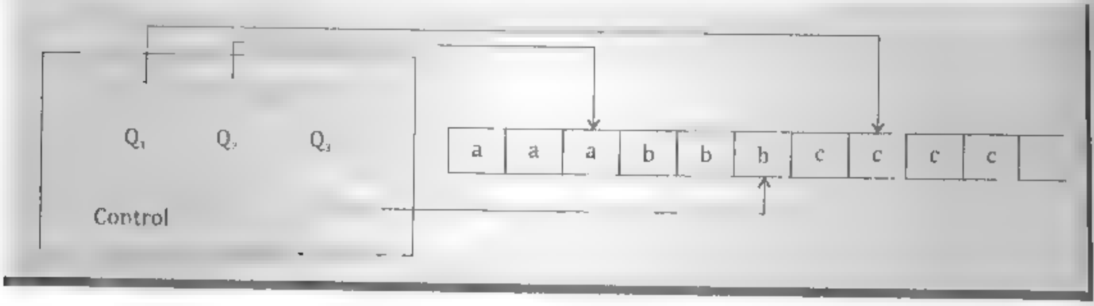
   Multi - dimensional TM has a finite control, a tape which consists of K-dimensional array of cells which is infinite in all 2K directions.

   Depending on the states and the symbol scanned, the device changes state, prints a new symbol, moves its head to one of the 2K directions either positively or negatively. Initially the input is along one axis and the head is at the left end of the input.

4. **Multi head TM**

   A multi head TM is a single tape TM with multiple read-write heads. However in every state only one head may be used. The heads are numbered from 1 to K. Therefore for K heads we have a partition of states Q_1, Q_2 .... Q_K where each Q_i contains the set of states which use the i^{th} head. A move of the TM depends on the state and on the symbol scanned by each of the head. In one move the heads may move independently to the left or to the right.

For K = 3, we might have something like this.



5.4 UNDECIDABILITY

A machine may accept a language or it may not accept (reject) a language. So the output of the machine may be accept or reject. This problem is called membership problem.

Definition

"Given a machine, M and a string x, does M accept x?" the output will be yes/no. Given a language, the machine may have to identify whether the language is finite or infinite. All such problems with two answers yes/no, accept/ reject, finite/infinite are called decision problems.

According to Church – Turing thesis, an appropriate way to formulate the idea of a decision algorithm precisely is to use a Turing Machine. For some decision problems we can write the algorithm for any specific instance, (which indicates that the problem can be solved using turing machine) but it is not possible to write a general decision algorithm that works for any given instance, which indicates that a turing machine does not exist to solve a decision problem.

That is there are problems that are solvable by TM and that are not solvable by TM, which can be divided into two groups, as

- The problems for which the solution exists in the form of algorithms. i.e., if there is an algorithm to solve a problem, there exists a TM that halts and results whether the input is accepted or rejected.
- The problems that run forever i.e , a TM will not halt on inputs that they do not accept.

5.5 RECURSIVE AND RECURSIVELY ENUMERABLE LANGUAGES

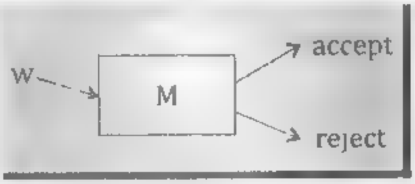
The decision problems that have decision algorithms with output yes/no are called solvable problems. According to Church-Turing thesis, an appropriate way to formulate precisely the idea of a decision algorithm is to use a Turing machine.

Recursive and recursively enumerable are the languages associated with turing machines. The language accepted by a turning machine, M is recursively enumerable if and only if L = L(M). i.e. if L is a language, there exists a turning machine that will halt and accept an input string in L and may either halt and reject or loop forever.

Definition

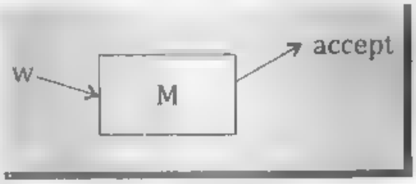
A language, L is recursive if L – L(M) for some Turing machine, M such that

- If *w* is in *L*, then *M* accepts.
- If *w* is not in *L*, then *M* eventually halts and never enters an accepting state



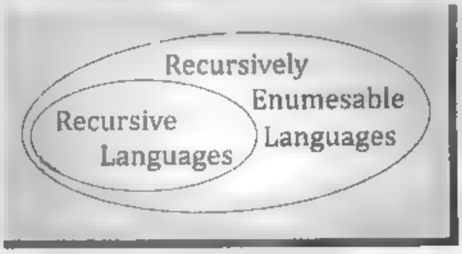
TM for a Recursive language.

A language, *L* is said to be recursive enumerable if there exists a turning machine that accepts it.



TM for a Recursively enumerable language.

i.e., *L* is recursively enumerable if *L* = *L* (*M*) for some turing machine, *M*. It doesn't say anything about *w* that is not in *L*. In other words if *L* is a language, there exists a turing machine that will halt and accept an input string in *L* or halt and reject otherwise.



Every recursive language is also recursively enumerable But it is not clear if every recursively enumerable language is also recursive.

Any instance of a problem for which the turing machine halts whether the input is accepted or rejected is called solvable or decidable problem. There are so many problems that are solvable. But there are some problems that are not solvable.

A problem whose language is recursive is said to be deicidable, else undecidable.

The problems that run forever on turing machine are not solvable. In other words, there are some problem input instances for which turing machines will not halt on inputs that they do not accept. Those problems are called unsolvable or undecidable problems. In general, if there is no general algorithm capable of solving every instance of the problem, then the decision problem is unsolvable. That is, a problem is undecidable if there is no algorithm that takes as input, an instance of the problem and determines whether the answer to that instance is yes or no.

5.5.1 Properties of Recursive and Recursively enumerable languages

The theorems which we are going to solve is proved by reducing one problem to another These reductions involve combining several turing machines to form a composite machine The state of the composite turing machine has a component for each individual component machine. Similarly the composite machine has separate tapes for each individual machine

When an algorithm is given, the composite turing machine performs one action if the algorithm accepts and another if it does not accept, we could not do this if we were given an arbitrary turing machine rather than an algorithm. Since if the turing machine did not accept, it might run forever, and the composite machine would never initiate the next task. In the pictures of the proofs, an arrow into a box labeled. "Start" indicates a start signal. Boxes with no "Start" signal are assumed to begin operating when the composite machine does. Algorithms have two outputs, "Accept" [Yes] and "Reject" [No] which can be used as start signals or as a response by the composite machine. Arbirary turing machines have only an "Accept" output, which can be used for the same purposes.

Closure properties of Recursive language

Recursive language are closed under the following operations. If L, L_1 and L_2 are recursive languages, then

- The Kleene closure L^* of L is recursive
- The concatenation of $L_1 . L_2$ is recursive
- The union $L_1 \cup L_2$ is recursive
- The intersection $L_1 \cap L_2$ is recursive
- The complement $L^1 = \sum^* - L$ is also recursive

Theorem 5.1

If L is a recursive language then its complement L^1 is also recursive

[or]

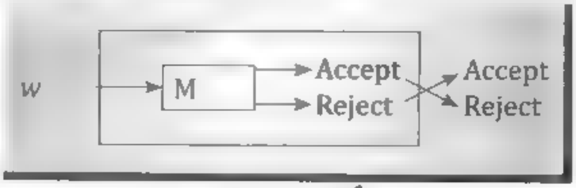
The complement of a recursive language is recursive.

Proof

Let L be a recursive language and M a turing machine that halts on all inputs and accepts L. Construct M^1 from M so that if M enters a final state on input w,

then M^1 halts without accepting. If M halts without accepting, M^1 enters a final state.

Since one of these two events occurs, M^1 is an algorithm. Clearly L (M^1) is the complement of L and thus complement of L is a recursive language.



construction of M^1 from M.

Theorem 5.2

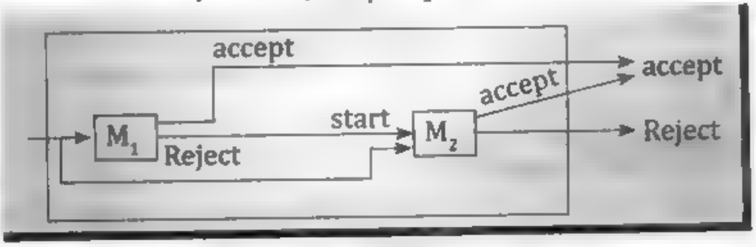
The union of two recursive languages is recursive.

OR

.he union of two recursively enumerable languages is recursively enumerable.

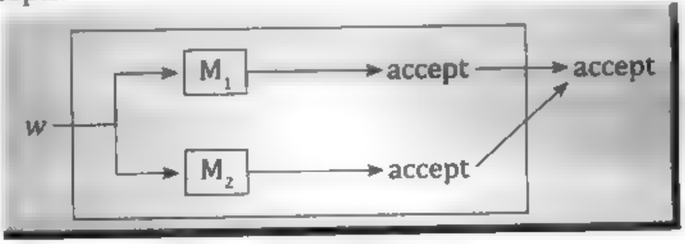
Proof

Let L_1 and L_2 be recursive languages accepted by algorithms M_1 and M_2 we construct M, which first simulates M_1. If M_1 accepts, then M accepts. If M_1 rejects, then M simulates M_2 and accepts if and only if M_2 accepts. Since both M_1 and M_2 are algorithms, M is guranteed to halt. Clearly M accepts $L_1 \cup L_2$.



Construction for union of recursive language.

For recursively enumerable languages the above construction does not work, since M_1 may not halt. Instead M can simultaneously simulate M_1 and M_2 on separate tapes. If either accepts, then M accepts.



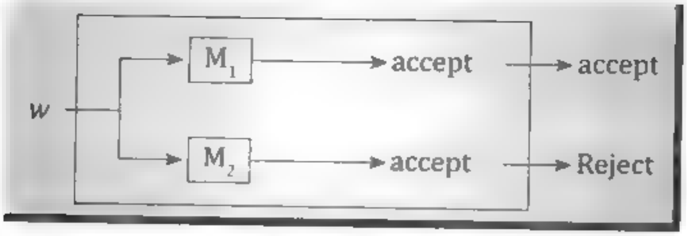
Construction of M^1 for recursively enumerable languages.

Theorem 5. 3

If a language L and its complement L^1 are both recursively enumerable, then L (and hence L^1) is recursive.

Proof

Let M_1 and M_2 accept L and L^1 respectively. Construct M by simulataneously simulating M_1 and M_2. M accepts w if M_1 accepts w and rejects w if $M2$ accepts w. Since w is in either L or L_1, we know that exactly one of M_1 or M_2 will accept. Thus M will always say either "accept" or "reject", but will never say both. Note that there is no priori limit on how long it may take before M_1 or M_2 accepts, but it is certain that one or the other will do so. Since M is an algorithm that accepts L, it follows that L is recursive.



construction of M for the theorem.

5.6 · Halting Problem of Turing Machine

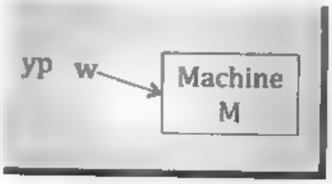
We say that problem A is reducible to problem B if a solution to problem B can be used to solve problem A. For example, if A is the problem of finding some root of $x^4 - 3x^2 + 2 = 0$ and B is the problem of finding some root of $x^2 - 2 = 0$, then A is reducible to B. As $x^2 - 2$ is a factor of $x^4 - 3x^2 + 2$, a root of $x^2 - 2 = 0$ is also a root of $x^4 - 3x^2 + 2 = 0$. If A is reducible to B and B is decidable then A is decidable. If A is reducible to B and A is undecidable, then B is undecidable. If a language L is not accepted by a turing machine, then the language is not recursively enumerable. Halting problem is an important problem which is not recursively enumerable that is unsolvable/undecidable decision problem.

The halting problem can informally be started as "given a turing machine M and an input string w with the initial configuration q_0, after some (or all) computations do the machine M halts?" In other words we have to identify whether (M, w) where M is the turning machine, halts or does not halt when *w* is applied as the input. The domain of this problem is to be taken as the set of all turing machines and all w. i e., given the description of an arbitrary turing machine M and the input string *w*, we are looking for a single turing machine that will predict whether or not the computations of M applied to w will halt. When we state decidability or undecidability results, we must always know what the domain is, because this may affect the conculsion. The problems may be decidable on some domain but not on another.

It is not possible to find the answer for halting problem by simulating the action of M on w by a universal turing machine, because there is no limit on the length of the computation. If M enters into an infinite loop, then no matter how long we wait, we can never be sure that M is in fact in a loop. The machine may be in a loop because of a very long computation. What is required is an algorithm that can determine the correct answer for any M and w by performing some analysis on the machines description and the input.

Given an arbitrary turing machine M = $(Q, \Sigma_{/n}, q_0, B, A)$ and the input

$w \in \Sigma^*$, does M halt on input w?



Theorem 5.4

The turing machine, M which halts on input w is undecidable.

Proof

We assume that the turing machine which halts [$HALT_{TM}$] is decidable and get a contradiction.

Let M_1, be the turing machine such that $T(M_1) = HALT_{TM}$ and let M_1 halt eventually on all (M, *w*).

We construct a turing machine, M_2 as follows.

Step 1: For M_2, (M, w) is an input.

Step 2: TM M_1, acts on (M, w).

Step 3: If M_1 rejects (M, w) then M_2 rejects (M, w).

Step 4: If M_1 accepts (M, w) simulate the TM M on the input string w until M halts.

Step 5: If M has accepted w, M2 accepts (M, w) otherwise M2 rejects (M, *w*).

In step 4 when M_1 accepts (M, *w*) the turing machine M halts on *w*. In this case either an accepting state *q* or a state *q* such that $\delta(q^1, a)$ is undefined till some symbol *a* in *w* is reached. In the first case M_2 accepts (M, w) and in the second case M_2 rejects (M, *w*).

It follows from the definition of M_2 that M_2 halts eventually.

Also T(M2) = {M, w)/The TM accepts w}.

$= A_{TM}$

This is a contradiction since A_{TM} is undecidable.

5.7 THE POST CORRESPONDENCE PROBLEM (PCP)

The post correspondence problem was first introduced by Emil post in 1946. Later, the problem was found to have many applications in the theory of formal languages. The problem over an alphabet Σ belongs to a class of yes/no [accept/reject] problems and is stated as follows:

Definition

Given two sequences of n strings on some alphabet Σ say

$A = w_1, w_2, \_\_\_\_ w_n$

and $B = v_1, v_2, \ldots\ldots v_n$

we say that there exists a post correspondence solution for pair (A, B) if there is a non empty sequence of integers i, j, ...k, such that

$w_i, w_j, \ldots\ldots w_k = v_i, v_j, \ldots\ldots v_k$

The post correspondence problem is to device an algorithm that will tell us, for any (A, B) whether or not there exists a solution. If there exists a solution to PCP, there exist infinitely many solutions.

Example :

Let $\Sigma = \{0, 1\}$

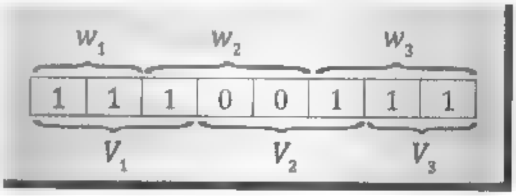
$A = \{w_1, w_2, w_3\}$

Where $w_1 = 11, w_2 = 100, w_3 = 111$

and $B = \{v_1, v_2, v_3\}$

where $v_1 = 111, v_2 = 001, v_3 = 11.$

For the pair (A, B) there is PCP solution as shown below.



If we take,

$w_1 = 00, w_2 = 001, w_3 = 1000$

$v_1 = 0, v_2 = 11, v_3 = 011$

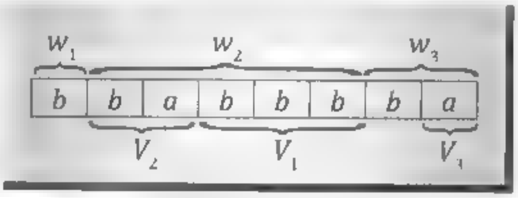
there cannot be any post correspondence solutions because any string composed of elements of A will be longer then the corresponding string from B.

PROBLEM 1

Does the PCP with two lists A = {*b*, *bab*^{3}, *ba*} and B = {*b*^{3}, *ba*, *ba*} have a solution.

SOLUTION

For the pair (A, B) there is a solution as shown below.



Short Answer Questions

1. Write the closure properties of recursive language.
2. Define post correspondence problem.

Long Answer Questions

1. Define TM, Language acceptance of TM and ID of TM
2. Define Recursive and Recursively enumerable languages
3. Explain halting problem of TM

MODEL QUESTION PAPERS

MODEL QUESTION PAPER 1

VI Semester BCA Degree Examination

(CBCS) (16-17 and onwards)

Computer Science

BCA 601T : Theory of Computation

Time 3 Hrs. **Max. Marks : 100**

Section A

Answer any ten questions. Each carries two marks **10 × 2 = 20**

1. Define $\in$ – NFA
2. List the algebraic laws governing regular expression.
3. Define nullable variables
4. Define CFG
5. Define Parse tree
6. List the properties of regular languages
7. Obtain a grammar to generate the set of all strings with exactly one a, over $\Sigma = \{a, b\}$
8. What are the ways to find the acceptance of a language by PDA
9. Find the language accepted by the following grammar.

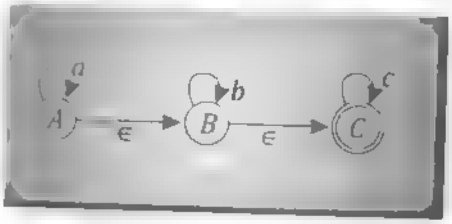
   $S \rightarrow aas|aa$

10. Define GNF
11. State the Halting problem of TM
12. State any two application of CFG.

Section B

Answer any five questions. Each carries 5 marks **5 × 5 = 25**

13. Convert the given ∈ – NFA



14. Show that regular languages are closed under intersection and reversal.
15 Construct an ∈ - NFA for the regular expression over $\Sigma = \{0, 1\}$ given by $(0 + 11)0^* 1$
16. Check whether the given grammar is ambiguous or not

$$S \to S + S \mid S \times S \mid a \mid b$$

17. Write short note on chomskey hierachy of languages.
18. Convert the following CFG to PDA

$$S \to aABC$$
$$S \to aB|a$$
$$S \to bA|b$$
$$S \to a$$

19. Eliminate left recursion from the following productions

$$E \to E + T \mid T$$
$$T \to T \times F \mid F$$
$$F \to (E) \mid id$$

20. Show that the complement of a recursive language is recursive

Section - C

Answer any three questions. Each question carries 15 marks. **3 × 15 = 45**

21.

a. Draw a DFA to accept strings of a's and b's having even number of a's and even number of b's.
7
b. Prove that if there exists NFA M_N which accepts the language $L(M_N)$, there exists an equivalent DFA such that $L(M_D) = L(M_N)$

22.

a. State and prove pumping lemma for regular languages. 8

b. Prove that $L = \{a^n b^n \mid n \geq 1\}$ is not regular

23. Define equivalent states and reduce the 10

a. DFA given below

	Q	a	b
*	1	3	2
	2	4	1
	3	5	4
	4	4	4
*	5	3	2

b. Compare DFA, NFA and $\in$ – NFA 5

24. Design a push down automa to accept the language

$L = \{wcw^R \mid w(\in(a+b)^*\}$ and check whether it is deterministic (10+5)

25. Simplify the following CFG 10

(a) $S \rightarrow aA \mid a \mid B \mid c$

$A \rightarrow aB \mid \in$

$B \rightarrow a\,A$

$C \rightarrow ccD$

$D \rightarrow abd$

(b) Describe the types of turing machine. 5

Section – D

Answer any One **1 × 1 = 10**

26. Convert the following CFG to GNF

$S \rightarrow AA \mid 0$

$A \rightarrow SS \mid 1$

27.

a. Describe post correspondence problem. 5

b. Construct a CFG to generate the set of all palindromes defined over {0, 1}

❖ ❖ ❖

MODEL QUESTION PAPER

VI Semester BCA Degree Examination

(CBCS) (16-17 and onwards)

Computer Science

BCA 601T : Theory of Computation

Time 3 Hrs. **Max. Marks : 100**

Section A

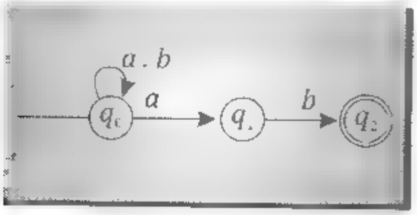
Answer any ten questions. Each carries two marks **10 × 2 = 20**

1. Define Symbol, Alphabet
2. Write the five tuples of FA
3. Design a DFA which ends with 01
4. Define ∈ – closure
5. Write the regular expression for a followed by zero or more number of *a*'s and *b*'s.
6. Define regular expression.
7. Define ID of a PDA
8. Explain nullable variable with an example.
9. Define recursively enumerable language
10. Write any four types of turing machine.
11. List the properties of CFG.
12. Define CNF.

Section B

Answer any fine questions. Each carries 5 marks **5 × 5 = 25**

13. Differentiate between DFA, NFA and ∈ – NFA
14. Convert the following NFA to DFA



15. Covert the following regular expression to ∈ – NFA
    $(a + b)^* ab (a + b)^*$

16. Obtain the string *aaabbabbba* by applying left most and rightmost derivation. Construct a parsetree from the following grammar

$$S \to aB \mid bA$$
$$A \to aS \mid bAA \mid a$$
$$B \to bS \mid aBB \mid b$$

17. Define TM and language accepted by a TM
18. Prove that the union of recursively enumerable language is recursively enumerable.
19. Eliminate left recursion from the given grammar

$$S \to Ab \mid a$$
$$A \to Ab \mid Sa$$

20. Write the general steps to covert CFG to PDA

Section C

Answer any three. Each carries 15 marks **15 × 3 = 45**

21.

a. Prove that following language is not regular $L = \{0^n \mid n \text{ is prime}\}$ 5

b. Minimize the following grammar 10

δ	0	1
A	B	E
B	C	F
* C	D	H
D	E	H
E	F	I
* F	G	B
G	H	B
H	I	C
* I	A	E

22.

a. Define parsing, parse tree, leftmost derivation and rightmost derivation. 8

b. Draw a DFA to accept strings of a's and b's ending with *abb* 7

23.

a. Simplify the following grammar 8

$S \rightarrow aAa$

$A \rightarrow Sb \mid bcc \mid aDA$

$C \rightarrow ab \mid aD$

$E \rightarrow ac$

$D \rightarrow aAD$

b. Write the steps to eliminate useless symbols. 7

24.

a. Construct a PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$ 8

b. Convert the following CFG to PDA 7

$S \rightarrow aABC$

$A \rightarrow aB \mid a$

$B \rightarrow bA \mid b$

$C \rightarrow a$

25.

a. Construct a TM to accept the language 10

L $= \{a^n b^n \mid n \geq 1\}$

b. Prove that if L and L1 are both recursively enumerable, then L and L^1 are recursive. 5

Section D

Answer any one. Each carries 10 Marks. **10 x 1 = 10**

26.

a. Draw a DFA to accept the language 5

$L = \{w : |w| \bmod 5 \neq 0\}$ on $\Sigma = \{a\}$

b. Define extented transition function for DFA 5

27.

a. Define CNF 2

b. Convert the following CFG to CNF 8

$S \rightarrow 0A / 1B /$

$A\ 0AA / 1S / 1$

$B\ 1BB / 0S / 0$

❖ ❖ ❖

60108637

ABOUT THE AUTHORS

Sarakutty T K is currently serving as Assistant Professor in the department of Computer Applications in Dayananda Sagar College, Bangalore. She has a rich teaching experience spanning over 19 years in both UG and PG students in the field of Computer Science. She was a member to the Board of examiners for PG-MCA & M.Sc Computer Science of Bangalore University and also a member of IDES.

T. Kohila Kanagalakshmi, M.C.A., M.Phil., M.E., Asst. Professor, MCA (BU) Dept, Dayananda Sagar College of Arts, Science and Commerce, has been in teaching profession for more than 7 years. She has secured gold medal in Master of Engineering from Anna University, Tamil Nadu and pursing research inthe area of data analytics in Pondicherry University, Tamil Nadu. She has qualified GATE Examination and presented research papers in IEEE conferences.

Skyward Publishers
157, 3rd Main, 7th Cross
Chamarajpet, Bengaluru - 18
Ph : + 91 80 2660 3535 / 6455 3333
Mob : + 91 96111 85999 / 85535 14780
Email : skyward.publishers@gmail.com
Website : www.skywardpublishers.com

ISBN : 978-93-84494-78-0

₹ 200-00